Remote monitoring of industrial drives with NB-IoT technology

Tatiana Kanerva

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 17.01.2018

Thesis supervisor:

D.Sc. (Tech.) Kalle Ruttik

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Tatiana Kanerva

Title: Remote monitoring of industrial drives with NB-IoT technology

Date: 17.01.2018 Language: English Number of pages: 7+77

Department of Communications and Networking

Professorship: Communication Engineering

Supervisor and advisor: D.Sc. (Tech.) Kalle Ruttik

The industrial process monitoring is an important issue, which has many applications. It allows to prevent failures of industrial equipment and to reduce maintenance costs. Monitoring processes are easy to deploy by using wireless technologies and such a deployment can result in lower costs. In this work, the application of the NB-IoT technology for industrial drive monitoring is analysed. ABB company, as a customer for this study, is interested in NB-IoT transmission of industrial drive data. In the context of this work, a literature search and a survey of NB-IoT chipset producers has been done to collect manufacture forecasts. At second, a test set *NETA-21 ARF* has been developed to measure the performance of Agile Radio Framework (ARF) for the industrial drive data transmission. Third, a demo set with an industrial drive has been built to validate the usage of ARF for industrial drive monitoring. Finally, the possible use cases of the NB-IoT technology for industrial process monitoring were illustrated.

The results of this work show that the NB-IoT technology can be used for the industrial drive monitoring. The survey of NB-IoT chipset producers discovered that the NB-IoT modules will be available in the turn of 2017 – in 2018 year. However, the current established price of the NB-IoT module is lower than the actual price expected by producers. The price range, which was specified by the chipset producers, is 7–10 $ while the established price of the module is 5 $. The ARF sends data with a transmission time of ≈200 ms for a low and ≈1.2 s for a high repetition factor. The ABB demo set performs the transmission of industrial drive data with the NB-IoT technology. The set allows to remotely observe condition of the drive. The use cases of NB-IoT technology for industrial process monitoring cover scenarios of dense production monitoring as well as distributed and mobile production monitoring.

Keywords: narrowband internet of things, industrial drive monitoring, industrial drive, NB-IoT chipset

Preface

I want to thank my supervisor Dr. Kalle Ruttik and Professor Riku Jäntti for their valuable guidance.

I am very grateful for the advice and clarification, which I have got from ABB representatives: Tapio Loponen, Zoltan Kosa, Mika J. Karna and Matti Kauhanen.

Also, I am very thankful for the explanations, clarification and help at any time from doctoral students Nicolas Malm, Yihenew Beyene and Keijo Lehtinen. Your advices helped me to understand what I should do and how I can perform it.

I want to thank Viktor Nässi. You are the person who gave me the possibility to obtain this topic for my Master's thesis.

There were teachers, during my studies in MSTU, who taught me a very important and extremely helpful topic – the engineering way of thinking. I want to thank Katsuba V.S., Vlasov A.B, Sorokin O.M. and Viskov A.J. for helping me to understand how an engineer should work.

Since I was a child, continuous support from my parents as well as their wonderful example guided me. Thank you, without your help I would not be able to study in Aalto university.

My dear sister Julia, Roman, and little Sasha: you gave me the strength to perform this study. I am forever thankful for you making my life so much full of light and happiness.

Finally, I want to thank my husband Dr. Mikko Kanerva. Your advice, your knowledge, your continuous care and support not only helped me with my studies but also strive me to develop myself to be a better person for you.

Otaniemi, 17.01.2018

T. Kanerva

Contents

Symbols and abbreviations

3GPP	The 3rd Generation Partnership Project
ADU	Applicational Data Unit
ARF	Agile Radio Framework
BS	Base Station
CPU	Central Processing Unit
DHCP	Dynamic Host Configuration Protocol
DL	Downlink
DRX	Discontinuous Reception
eDRX	extended Discontinuous Reception
eNB, eNodeB	E-UTRAN Node B
eNB-Ca	eNB C++ application
eNB-Js	eNB Javascript application
eNB-Ps	eNB Python script
FDD	Frequency Division Duplex
GSM	Global System for Mobile Communications
HARQ	Hybrid Automatic Repeat Request
IoT	Internet-of-Things
IP	Internet Protocol
LTE	3GPP Long Term Evolution
LWPA	Low-Power Wide-Area
MIB	Master Information Block
MTC	Machine Type Communication
NB-IoT	Narrowband internet of things
NPBCH	Narrowband Physical broadcast channel
NPDCCH	Narrowband Physical Downlink Control Channel
NPDSCH	Narrowband Physical Downlink Shared Channel
NPRACH	Narrowband Physical Random Access Channel
NPSS	Narrowband Primary Synchronization Signal
NPUSCH	Narrowband Physical Uplink Shared Channel
NRS	Narrowband Reference Signal
NSSS	Narrowband Secondary Synchronization Signal
OFDMA	Orthogonal frequency-division multiple access
PC	Personal Computer
PDU	Protocol Data Unit
PRB	Physical Resource Block
PSM	Power Saving Mode
RACH	Random Access Channel
RAM	Random Access Memory
RAT	Radio Access Technology
RAR	random access response
RF	Radio Frequency
SBC	Single Board Computer
SC-FDMA	Single-carrier Frequency Division Multiple Access
SDR	Software Radio
SSH	Secure Shell
TAU	Tracking Area Update

TUN	Network TUNnel
UDP	User Datagram Protocol
UE	User Equipment
UE-Ca	UE C++ application
UE-Ps	UE Python script
UL	Uplink
USRP	Universal Software Radio Peripheral

1 Introduction

Machine Type Communication (MTC) has become a very popular topic recently. The growing amount of various devices and sensors requires development of new technologies, which allow to support massive number of devices and are optimized for data transmission between heterogeneous devices [1, 2].

One of the applications of MTC is the monitoring of an industrial process. This monitoring does not require multiple handovers. However, there could be a significant amount of sensors, which usually are located in places difficult to access and, possibly, in places where mobile coverage is poor.

The monitoring of the industrial process could be done in different ways. In this work, non-real time monitoring is considered. This means that the sensor parameters are collected and then centrally analysed with a purpose to optimize the industrial process. This kind of monitoring has no strict requirements for the time delay. Hereby, a technology, which suits the requirements for the described type of monitoring, could be Narrowband Internet of Things (NB-IoT) [3].

NB-IoT is a new cellular radio interface defined in The 3^{rd} Generation Partnership Project (3GPP) Release 13. NB-IoT is optimized for machine type communication, where one E-UTRAN Node B (eNodeB, eNB – a base station in network standard 3GPP Long Term Evolution) could support a significant amount of User Equipment (UE) pieces. For this standard, the UEs are placed stationary. Typically, they send a small amount of data to the eNodeB. The data transmission for NB-IoT is not delay sensitive [4, 5].

In this thesis, I will study the suitability of NB-IoT for remote monitoring and control of industrial drives. To perform the study, I will (1) interview NB-IoT chipset producing companies and carry out a questionnaire to find out NB-IoT chipset availability, price, value chain, eSim support and feature set, (2) analyse the performance of NB-IoT system represented by Agile Radio Framework (ARF) for transmission of industrial drive data, and (3) describe use cases for monitoring of an industrial process with NB-IoT.

The description of the NB-IoT technology and ABB solutions can be found in the Background section. Research methodology and Systems build-up sections contain the Master's thesis task description and the solution procedure for the problems. The outcomes of the work are given in the Results section. Finally, the main conclusions can be found in the section Summary.

2 Master's thesis problem description

The main goal of this thesis is to evaluate the applicability of the NB-IoT technology for monitoring the industrial drives using a NETA-21 tool. Therefore, the following subtasks were established:

1. The surveying of the NB-IoT chipset producers and gather the following information about the NB-IoT modules:
   - release time of NB-IoT modules;
   - price of NB-IoT modules;
   - eSIM support option;
   - the value chain;
   - feature set of NB-IoT chipsets.
2. The measuring of the performance of the NB-IoT for data transmission by NETA-21 monitoring tools.
3. The development of a demo to verify the implementation of the ARF system for industrial drive data transmission.
4. The determination of scenarios for the NB-IoT system usage while monitoring of the industrial drives.

2.1 Master's thesis overview

This thesis is organized as follows. Section 3 contains fundamental information about the technologies, equipment and systems, which have been used for this Master's thesis. In Section 4.1, can be found information about the chipset producers questionnaire and Section 6.1 contains the list of NB-IoT chipset producers and the questionnaire results. The section 5.1 contains description of the hardware set, software implementation and principles of operation of a measurement set, which has been developed for performance analysis of data transmission from the NETA-21 monitoring tool by ARF. In turn, Section 6.2 contains the results of the performance measurement. Section 5.2 includes information about a demonstration set for validation of the operation of ARF for the data transmission of the industrial drives, including software and hardware components description and principles of operation. Section 6.3 contains information about the development of the demo set, reforming of ARF software side and assembling of the hardware set. In Section 4.2, principles of development of use cases for NB-IoT usage for industrial process monitoring are described. Section 6.4 includes the description of possible use cases. Finally, Section 7 contains the brief description of the thesis work and the conclusions.

3 Background

3.1 Narrowband Internet of Things

3.1.1 Machine Type Communication

Nowadays, we can see a significant growth in the amount of various devices. We use them for many different purposes to optimize work, to minimize human involvement in processes, which could be automated, and, of course, to reduce costs. The devices could be different sensors for industrial and home purposes, various instruments, household machines, healthcare equipment, etc.

Considerable amount of devices require methods for communication: to transfer data to a control center, receive commands and, if required, communicate with other devices.

The wiring multiple devices is difficult and costly, therefore, there is interest to connect devices over wireless network. However, contemporary wireless networks are mostly developed for high-speed transmission of a big amount of data. In the case of machine communication, we have other kinds of requirements. The requirements of MTC highly depend on the considered use case. However, the general requirements are:

- keeping the technology deployment and the equipment simple and low-cost;
- consuming of energy have to be very low;
- support of massive number of sensors;
- reliability and availability of the technology.

The existing mobile communication systems could be tailored for machine type communication by extension of coverage, reduction of UE complexity, enabling support of massive number of UE and the extending of battery lifetime [6, 7]. Other features of the MTC technology are specified based on an use case.

For example, if we consider a car tracking system, we would need wireless transmission technology, which supports a significant amount of devices, can perform multiple handovers, and works with numerous delay sensitive transmissions of small packets.

Another example is the collecting of data from in house sensors. In this case, sensors are placed stationary and handovers are almost not needed. The sensors send small packets of data once per month. This transmission is not delay sensitive. But the amount of supported devices could be massive [4]. Such type of a communication is typically called Low-Power Wide-Area (LWPA) networks.

In this work, I consider an use case of MTC communication where industrial process equipment sends data packets of at a maximum of 1 kB size to a central cloud. The sensors are used for monitoring and analysis but not for the equipment control and as such the transmission is not delay sensitive. A suitable technology for such use case could be the recently introduced Narrowband Internet of Things.

3.1.2 The 3GPP and the NB-IoT

The 3^{rd} Generation Partnership Project (3GPP) unites seven telecommunications standard development organizations. The 3GPP members produces the Reports and Specifications that define 3GPP technologies. This project provides complete system specifications for cellular telecommunications network technologies.

NB-IoT is a new cellular technology for low rate data transmission first introduced in 3GPP Release 13. NB-IoT is designed to provide wide area coverage for the Internet of Things (IoT) [6]. To meet the cost reduction and energy saving requirements of MTC, NB-IoT is kept as simple as possible.

Energy efficiency is an important requirement for MTC and in the standard a lot of attention is paid for it. According the specifications, device battery lifetime should be longer than 10 years.

NB-IoT uses a licensed spectrum and supports three different types of deployment, which allow the coexistence of the technology with present mobile communication systems.

NB-IoT has a peak downlink (DL) rate of 170 kbps and uplink (UL) rate of 250 kbps [8] It uses Frequency Division Duplex (FDD) half duplex for data transmission. It means that the downlink and uplink are separated over frequency and the communication between UE and eNodeB cannot be performed simultaneously [4]. The NB-IoT bandwidth is the same for downlink and uplink − 180 kHz, therefore it could use for the transmission one Global System for Mobile Communications (GSM) carrier, which is 200 kHz, or it could be included in an 3GPP Long Term Evolution (LTE) carrier. The system supports only one antenna. The UE transmission power is 20/23 dBm, the target uplink latency is 10 seconds [4, 7].

3.1.3 The deployment of NB-IoT

Effective deployment scenarios are one way to reduce the costs of new technologies. Introduction of NB-IoT can be done via a centralized software upgrade, which will deploy the new technology connectivity on top of the existing cellular network [8].

NB-IoT has been created such a way that it could coexist with present cellular systems. The downlink of NB-IoT is based on Orthogonal Frequency-Division Multiple Access (OFDMA) with a 15 kHz subcarrier spacing. The uplink, which supports multi-tone and single-tone transmissions, is based on Single-carrier Frequency Division Multiple Access (SC-FDMA). The multi-tone transmission has a subcarrier spacing of 15 kHz, while the single-tone transmission supports a 15 kHz and 3.75 kHz subcarrier spacing. NB-IoT technology in turn requires a frequency band of 180 kHz (for uplink and for downlink). These parameters of the technology allow coexistence with LTE [6].

Thereby, the implementation of an NB-IoT carrier to the present system could have the following scenarios (see Figure 1) [6, 7]:

1. standalone;
2. in-band;
3. guard-band.

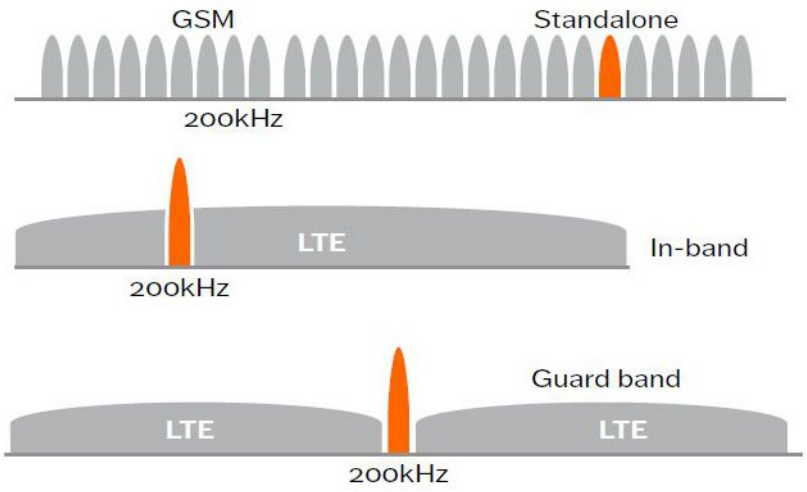


Figure 1: Different deployment scenarios of NB-IoT. Modified from [7].

For the standalone deployment option, a GSM carrier is replaced with an NB-IoT carrier. The GSM carrier bandwidth is 200 kHz while NB-IoT requires 180 kHz bandwidth. Therefore, in the case of standalone deployment, there is in addition a guard interval of 10 kHz on both sides of the spectrum [4].

The in-band scenario implies the use of a part of an LTE carrier. This method provides the most spectrum and cost-efficient deployment of NB-IoT for operators with mostly LTE spectrum available [7]. In this case, NB-IoT uses one physical resource block (PRB) - elementary unit of resource allocation, which can be used by LTE whenever NB-IoT does not need it [7].

The third deployment scenario is similar to the in-band deployment but, in this case, NB-IoT carrier is placed in the guard-band of an LTE carrier.

3.1.4 NB-IoT Physical Channels

NB-IoT supports three physical channels and two physical signals for the downlink. The channels are [4, 6]:

1. NPBCH – the narrowband physical broadcast channel – the channel carries the master information block (MIB). MIB contains the necessary information for an UE such as downlink Bandwidth, Number of Transmit Antenna, Reference Signal Transmit Power, etc. After receiving MIB, the UE can connect to the network.

2. NPDCCH – the narrowband physical downlink control channel – this channel carries scheduling information for uplink and downlink channels, Hybrid Automatic Repeat Request (HARQ) acknowledgement, paging indication and Random Access Response (RAR) scheduling information.

3. NPDSCH – the narrowband physical downlink shared channel – the channel carries data from the higher layers, paging message, system information and the RAR message.

The physical signals of the DL are [6]:

– NPSS – Narrowband Primary Synchronization Signal;

– NSSS – Narrowband Secondary Synchronization Signal;

– NRS – Narrowband Reference Signal.

NPSS and NSSS are used for time and frequency synchronization as well as cell identity detection [6]. NRS provides phase reference, which is needed for the demodulation of the DL channels [6].

In the UL, NB-IoT supports two physical channels and one physical signal [4, 6]:

1. NPRACH – Narrowband Physical Random Access Channel – the channel is used for Random Access Channel (RACH) transmission. RACH is the initial message from UE to eNodeB after the UE is powered on.

2. NPUSCH – Narrowband Physical Uplink Shared Channel – this channel supports two formats:

   – Format 1 is used for carrying UL data. The transport blocks of Format 1 is not bigger than 1000 bits.
   – Format 2 carries UL control information. It is used for sending the HARQ acknowledgement.

3. DMRS – Demodulation Reference Signal – the signal assists coherent demodulation.

3.1.5 NB-IoT Random Access Procedure

Random Access in NB-IoT implies both an initial access of a UE after it has been powered on and a scheduling request [6]. RACH main purposes are to achieve UL synchronization and to obtain the resource for the transmission. The random access procedure of NB-IoT is similar to LTE. It includes four steps:

1. At first, UE sends a random access preamble.

2. After that, eNB transmits a random access response. It contains the timing advance command and scheduling of uplink resources.

3. At third, UE performs scheduled transmission.

4. At last, eNB sends to the UE a connection resolution. The resolution is needed to solve a problem when several UEs send the identical random access preambles.

The scheme of the Random Access Procedure is shown in Figure 2

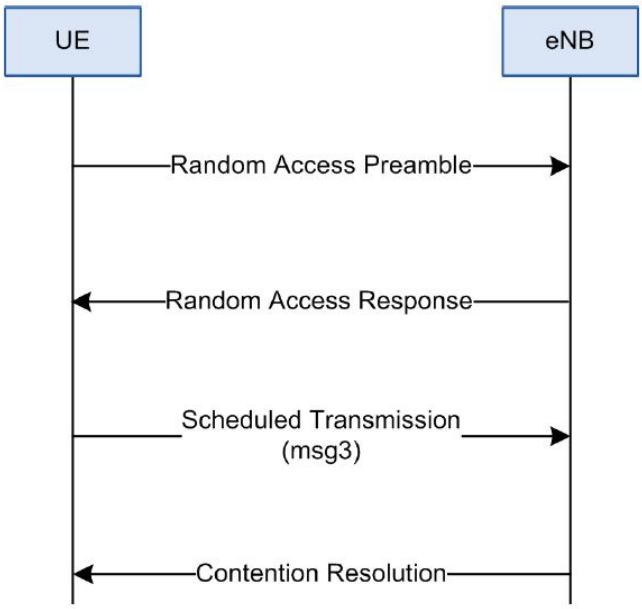


Figure 2: The Random Access Procedure [4].

3.1.6 The coverage extension

The use case of NB-IoT implies collecting data from sensors, which are usually placed in location that are difficult to access such as basement floors or places in rural areas. Typically, such places have also bad coverage of the mobile network. Therefore, NB-IoT is designed to provide additional 20 dB to link budged compared to LTE networks. The link budged improvement corresponds to 7–10 times better coverage area, or, coverage of loss caused by signal propagation through walls of a building [7, 8].

NB-IoT achieves the coverage extension by increasing of number of message repetitions. The main advantage of this technology is low complexity, although, in turn, the high repetition factor decreases the data rate.

NB-IoT provides up to three coverage enhancement (CE) levels (level 0 – level 2) for NPRACH to serve UEs, which have different coverage conditions [4, 6]. Level 0 implies the normal coverage while level 2 refers to the worse coverage case [4]. The network defines how many CE levels are defined and how NPRACH resource should be configured for every level. For each configuration, a repetition value of random access preamble is defined. The number of repetitions could vary from 1 to 128 times. The repetition number can be only a power of 2. UE uses the same transmission power on each transmission [4].

The determination of NPRACH resources goes as follows. UE measures the downlink signal power and defines the coverage level [6]. After determination of the coverage level, the UE transmits random access preamble using the NPRACH resources configured for its coverage level [4, 6].

Before each NPUSCH transmission UE receives control information through NPDCCH, where repetition number is specified among other control information, which is necessary for the transmission. In general, NB-IoT implies repetition for each channel [9].

NPUSCH provides a data rate of 20 bps for 128 repetition factor and the NPDSCH provides a data rate of 35 bps for 512 repetition factor with the lowest modulation and coding scheme [6].

3.1.7 The reduction of UE complexity

The generally specified NB-IoT module price must be less than 5 \$ ($\approx$ 5 €) [7]. To keep the price of the device low, it is required to decrease the complexity of a NB-IoT device. The NB-IoT device can be a chipset or an external module, which can be connected to any equipment through the existing interfaces. The following techniques have been applied to make the modules less expensive [8]:

- half duplex FDD transmission;
- single receive chain;
- low throughput;
- low Radio Frequency (RF) bandwidth;
- lower power class of a device.

In addition, there are number of other low cost technologies beyond the 3GPP standardization, which could allow industry to reduce the price of the module. The key features are a high level of component integration and a limited usage of Radio Access Technologies (RAT) and RF bands.

3.1.8 NB-IoT capacity and latency

NB-IoT capacity target is 40 devices per household. To meet the requirements, NB-IoT uses only one PRB in UL and in DL [6]. In addition, a UE can use a single subcarrier NPUSCH in UL [6].

According [6] traffic model shows that NB-IoT with one PRB supports more than 52500 UEs per cell. In [7] is mentioned that the simulation of NB-IoT system show support of 200 000 devices for a cell. This is four times more than the specification targeted [7].

NB-IoT latency target is less than 10 s. As NB-IoT system handles many devices simultaneously, it has to provide connectivity to all of them irrespective to their coverage conditions. Thereby, an important technique for NB-IoT is the multiplexing of traffic of UEs [7].

For the devices placed in coverage area with bad signal quality, the providing of satisfactory data rate can be achieved by concentrating their transmission power to a narrow bandwidth. This can be done by using single OFDM subcarrier instead of all subcarriers in a resource block. NB-IoT subcarrier bandwidth can be 15 kHz or even 3.75 kHz. In the case of 15 kHz bandwidth, the NB-IoT has the slot duration 0.5 ms, the subframe duration 1 ms and the frame duration 10 ms – the same as in LTE. If the subcarrier bandwidth is 3.75 kHz, the NB-IoT slot duration is made 2 ms to preserve the coexistance with LTE [6, 7].

To increase the data rate, more bandwidth can be allocated for the communication. According [7], data rates could be increased up to 12 times if, instead of single tone, multi-tone transmission would be used. However, this method works good only if there are many sensors at a good coverage area [6, 7].

In Table 1 can be found information from [7] about uplink latency values. In the table presented maximum uplink latency for an NB-IoT device placed in coverage areas with different signal level. We can see that the latency grows significantly if the device has bad signal quality, but even the worst case of signal coverage (+20dB) is still under the specified 10 seconds.

Table 1: Latency table of NB-IoT [7].

Coverage	**Sync**	**MIB**	**PRACH**	**RAmsg2-4**	**ULgrant**	**ULdata**	**Ack**	**Uldata**	**TOTAL**
+0 dB	340	151	324	622	48	39	41	39	1,604
+10 dB	340	151	688	708	45	553	47	553	3,085
+20 dB	520	631	1,440	1,060	49	1,923	77	1,923	7,623
Duration (ms)									

3.1.9 The battery lifetime

The UEs of the NB-IoT system are usually located in hard-to-reach places where there is no direct power source, therefore the devices are powered from battery. Moreover, the battery replacement is difficult and costly procedure so the battery lifetime should be comparable with the lifetime of the devise.

The targeted battery operation period for NB-IoT is 10 years. To achieve this target, the technology should utilize UE idle and sleep modes.

In the Release 13 of 3GPP an enhanced Discontinuous Reception (eDRX) was introduced. For the normal Discontinuous Reception (DRX), the device supports two modes: DRX_connected and DRX_idle. In the idle mode, the device turns off the receiver and checks for paging messages periodically. In the connected mode, if there is no either UL or DL transmission, the device goes to DRX mode. It means that it does not monitor NPDCCH in every subframe but checks it discontinuously [10]. The difference between DRX and eDRX is how often the UE checks if the eNB has something to transmit – the DRX cycle duration. For DRX, it is 2.56 seconds, while for eDRX it is extended to 10.24 seconds [7].

Another technique for power saving is introduced in the 3GPP release 12 Power Saving Mode (PSM): after the device moved to the idle state, it starts a timer, which specifies how long UE continues to check for paging in order to stay reachable for the network. Once the timer expires, the device goes to PSM and, while it is in this mode, it does not listen for paging. This means that it cannot be reached by the network. The UE stays in PSM until it has to make any periodic transaction, for example, Tracking Area Update (TAU) or it has to transmit some data to the eNB [8].

3.2 Agile Radio Framework

In this work, an implementation of NB-IoT on basis of software defined radio (SDR) is used to perform the tasks of the Master's thesis. The system has been developed by researchers from the Communication and Network Department of Aalto university. It is called Agile Radio Framework – ARF.

ARF corresponds to a NB-IoT protocol stack written in C++. The system can deliver Internet Protocol (IP) packets, which have been feed to it over a TUNnel (TUN) interface available in Linux platform.

The TUN interface is a software-based interface, which is seen by the kernel as a real physical device. TUN interface transmits and receives packets for a user space application. The interface works as a normal Point-to-Point or Ethernet device. It receives packets from a user space program, while the real physical device receives them from physical media. Similar, TUN writes the packets to the user space program but not to the physical media [11].

The TUN interface is a convenient solution because the data written to a TUN interface is automatically converted to an IP packet. Hereby, there is no need to create the IP packet, it can be just taken from the TUN and routed further.

At the software level, ARF consist of two C++ applications. One of the applications performs UE functionality, in this work it is called UE-Ca (UE C++ application). Another one works as a NB-IoT eNB, and in this work it is called eNB-Ca (eNB C++ application).

The following hardware is required to ran ARF:

1. Two laptops: one for UE side and another one for eNB side.

2. Two Universal Software Radio Peripherals (USRPs), which perform wireless transmission of packets.

The ARF works at NB-IoT standalone mode and uses a GSM carrier for the data transmission.

To present the performance of ARF, a demo platform has been developed. The scheme of the demo platform can be found in Figure 3

ARF demo platform executes reading of data from a sensor, creates a data packet, feeds it to the TUN interface where the packet is taken by ARF NB-IoT UE implementation and transmitted to the eNodeB side.

Figure 3: ARF operation. (Courtesy T. Kanerva)

The sensor data is read by a Python script. In this work this Python script is called UE-Ps (UE Python script).

At the eNodeB side, the packet is taken by eNB-Ca and then redirected to an application, which recovers data from the packet. The application is a Python script, which is called in this work eNB-Ps (eNodeB Python script).

Finally, the decoded data are represented at the eNode side with a Javascript application (eNB-Js – eNodeB Javascript).

3.2.1 Principles of operation

AT the beginning of ARF operation, the NB-IoT UE synchronises with the NB-IoT eNodeB. Once it is has been done, the data connection between these two nodes will be established.

The UE-Ps obtains data from a sensor, creates an User Datagram Protocol (UDP) packet and sends it to the UE-Ca by feeding it to a configured TUN interface. Then UE-Ca sends the UDP packet to eNodeB through USRP transceiver. At another end, USRP transceiver receives the packet. The eNB-Ca extracts the data and presents it with a web-based user interface. The user interface is ran by the eNB-Ps.

In the event of failed transmission, the packet are simply resent [12].

3.2.2 The transmission of the industrial drive data with ARF

ARF has been chosen to analyse NB-IoT performance for the transmission of the industrial drive data. For this purpose, the industrial drive is considered to be a sensor. The UE-Ps reads the parameters from the drive and sends them via UE-Ca and eNB-Ca to the eNB-Ps for the presentation (see Figure 4). After NB-IoT chipsets will be in mass production, the UE side could be carried out by hardware equipment.

To make the system smaller and easy for transportation, the laptops, which run the ARF code, can be changed to the single board computers (SBC). The choice of SBC should be made carefully. ARF has quite relative high performance requirements, so the computer should have enough powerful processor and, at least, four cores. Another requirement for the SBC is that it would have a real Ethernet connector but not an Ethernet port, which is connected to the board through a USB bridge. Usage an USB bridge decreases speed of the transmission, therefore, the system does not meet the latency requirements. For the same reason, the system cannot be ran through the USB-Ethernet adaptor.

Figure 4: The adaptation of ARF for the transmission of industrial drive data. (Courtesy T. Kanerva)

3.3 The ABB technologies

ABB company provides various equipment called industrial drives, which are used to drive production processes. A monitoring tool NETA-21 developed in ABB could be used to get information about the industrial drive condition. The NETA-21 sends the collected from industrial drives data to a cloud, where this information is analysed. Data analysis could predict an industrial drive failure or it could develop a better usage scenario of the drive, which will allow more optimal and effective production. The service also sends notifications about drives breakdowns to a customer email or phone and possibly, in future, this technology could carry out the industrial drives control.

In this work we are interested in NB-IoT system to perform data transmission from the NETA-21 tools to the cloud.

3.3.1 Industrial drives

Any automated industrial process must move, in other words, there should be a motor, which converts some kind of energy to the motion of the industrial equipment. The energy, which is needed to be converted to motion by the motor, can be electrical energy. Then the motors are called electrical motors. Nowadays, electrical motors can be used in almost whatever production: moving of conveyor belts or escalators, pumping of water, robots' motion, etc.

An electrical motor could run at different speed. The higher speed of the motor the faster the industrial process can be impacted. Normally, the deployment demands speeds from high to slow. If the motor runs all the time at the highest speed, the industrial process control is not optimal. Therefore, there is a need for an equipment, which allows to vary speed of the motor. An industrial drive is a machinery, which allows to run the motor at the different speed depending on requirements of the industrial process.

The benefits of using industrial drives are [13]:

1. Energy savings — if the motor speed can be reduced without harming the industrial process, it would lead to the significant energy saving and, as a result,

it allows to decrease the production costs.

2. Optimal process control — the accurate control of the motor speed and torque would allow to increase the quality and throughput of the final product.

3. Reduced need for maintenance — the ability to vary the motor speed and torque reduces tear and wear of the motor and the driven equipment. For example, increasing of motor speed slowly up to the required value prevents sudden shock loading, which can damage elements of the motor and the driven equipment over time.

4. Efficient system upgrade — the drive allows for removal replaceable elements. In addition, the drive provides lower starting current, this allows for growth of number of industrial equipment.

5. Functional safety — the drive can provide functional safety features, this ensures safety for the operators of the industrial equipment.

3.3.1.1 The Industrial Drive Parameters

Industrial drives use a set of parameters. The parameters represent the performance of the industrial drive, set reference and control values. The parameters of the industrial drive are united into the parameters groups. For example, the parameters *01.01 Motor speed used*, *01.06 Output frequency*, *01.07 Motor current*, *01.10 Motor torque %* belong to the parameter group *01 Actual values* and the parameters *22.01 Speed ref unlimited 22.11 Speed ref1 selection*, *22.13 Speed ref1 function* belong to the parameters group *22 Speed reference selection*.

The parameter groups are ordered and have own constant number as the parameters they contain. The number of the parameters group and the the actual number of the parameter specify the address where the value of the parameter is stored in memory.

In this thesis, the parameters groups *01 Actual values*, *20 Start/stop/direction* and *22 Speed reference selection* are considered mostly. If the group *01* represents the condition of an industrial drive, then groups *20* and *22* specify values for the industrial drive control [14].

3.3.1.2 The ACS880-01 democase

The ABB company provides democases, which help to investigate the principles of operation of the industrial drive. The picture of ACS880-01 democase, which is used for this thesis can be found on Figure 5.

The ACS880-01 democase consists of a board where are placed an ACS880 series industrial drive, a small motor with a mechanical break and two control panels. One control panel is a control panel of the industrial drive placed at the board and other one is an additional control panel, which contains knobs and buttons, which are convenient for the drive management. The demo platform operates such that the

Figure 5: The ACS880-01 democase.

industrial drive must be connected to the control panels with an Ethernet cable. Without this connection, the democase cannot be started.

This democase was used to build the demo set, which presents the performance of NB-IoT system for the transmission of the industrial drive data.

To connect the industrial drive to ARF, the Panel bus connection was used. The same connection is used to perform communication of the industrial drive with the control panels. As we did not have a switch for this type of connection, the democase must be turned on with the connected control panels at first and then the cable should be taken from the control panels port and connected to the system [15].

The democase has the limited feature set in comparison with the real industrial drive. For example, it does not have an XD2D connector, which is used for the Fieldbus control. The absence of this connector resulted in the lack of ability to write the parameters to the drive of the democase. Therefore, the writing of the parameters to the drive is represented only theoretically in this thesis.

3.3.2 The NETA-21 monitoring tool

The NETA-21 is a monitoring tool for industrial drives provided by ABB company. NETA-21 contains different access interfaces for a wide range of ABB drives. The monitoring tool performs the collection of the industrial drive data and provides for a user a web interface to access the equipment remotely. Thereby, NETA-21 allows to perform the maintenance of the industrial drives remotely.

The main features of NETA-21 are [16]:

- collection of the industrial drive parameters and their storage;
- providing of access to the parameters of the industrial drives, their event list and data logs through a web interface;

– configuration of the user access levels and secure access protocols;

– notification about events, warnings or faults of the industrial drives;

– providing the remote services.

In this work, most of the attention is paid to the scenario when a NETA-21 monitoring tool sends collected from the industrial drives data to a cloud.

The average size of a sample received from a single drive is 100 bytes. The maximum amount of the drives connected to the NETA-21 is ten due to restrictions of data collection throughput. Thereby, the maximum size of the collected data packet is around 1 kB [17].

After thereceiving, the collected data is assembled to an XML file. The file could be compressed to a ZIP archive before sending. If the compression is not done, the encoding is needed to be applied to the data [17].

The intervals of the data collection and the packet transmission to the cloud can be specified. This significantly affects requirements for the throughput. Therefore, in this work, I will try to find out the limitations, which NB-IoT system brings for the data transmission from NETA-21 monitoring tool to a cloud.

3.3.2.1 Accessing of NETA-21 monitoring tool

The detailed instruction of how to access NETA-21 monitoring tool can be found in [18].

The access to the monitoring tool is performed through a web interface or with Secure Shell (SSH). At first, a user must initiate the functioning of the NETA-21 as a Dynamic Host Configuration Protocol (DHCP) server via the ETH1 port. To make this, the user must press *SD RJ45* button for 5 seconds. After the DHCP server is started, the user can access the NETA-21 with a browser by navigating to `http://192.168.230.1` or by SSH to the user **factory** [18].

The passwords for SSH access are different for every NETA-21 monitoring tool. For the tool, this work has been performed with, the SSH password for the user **factory** is **Bj6X!4K5BC** and the password for the root access is **e2C@3UJhwg** [17].

3.3.2.2 The NETA-21 monitoring tool as a Modbus/TCP gateway

The NETA-21 monitoring tool can be used as a Modbus/TCP gateway. It allows to read the parameters of the industrial drives by a Personal Computer (PC) connected to NETA-21 with a typical TCP/IP connection.

To turn on this function, the Firewall Settings for Modbus/TCP gateway must be set up. For more detailed instruction see Appendix D.

MODBUS protocol supports at maximum 247 devices. For every connection type there is a range of ID numbers for the drives connected to the NETA-21. The ID ranges for the different protocols can be found in Table 2 [19].

The ID of a specific driver can be found in the web interface. It is coded inside the internal name of the drive. The internal name can be seen in **Device parameters**

Table 2: Ranges of the device IDs for different protocols supported by NETA-21.

Protocol	**ID Range**
DDCS fiber optics	1 – 32
Panel bus 1 (port PNL1)	33–64
Panel bus 2 (port PNL2)	65 – 96
Modbus/RTU	97 – 128
Ethernet tool network	129 – 160
Reserved	161 – 247

panel under **Device interfaces**. The last number in the name is the device index inside the range of the used protocol [19].

For example, if only one drive connected to the first Panel bus port of NETA-21 (port PNL1), the device name is *Panel bus 11* and the device ID is 33.

As has been described in Section 3.3.1.1, the performance of an industrial drive is presented by a set of parameters collected to groups. The address of the specific parameter must be calculated by Formula 1.

$$ParameterAddress = 40000 + 100 \cdot GroupNumber + ParameterNumber - 1 \quad (1)$$

Historically, the beginning of parameters numbering in MODBUS protocol starts from 40000. Nowadays, the most of Modbus clients automatically omit this space, thereby, the calculation of the address of the specific parameter must be performed with Formula 2.

$$ParameterAddress = 100 \cdot GroupNumber + ParameterNumber - 1 \quad (2)$$

For example, if we would like to read the 6^{th} parameter in the first group with the client, which omits the 40000 space, the request must be sent to the address

$$ParameterAddress = 100 \cdot 1 + 6 - 1 = 105$$

.

The reading of the parameters is performed with MODBUS function code 03 *Read Holding Registers*. **NETA-2 Firmware 2.23** also supports the *read/write* option [19]. For more information about reading and writing the parameters, see Appendix D.

MODBUS protocol returns for the request a raw 16 bit value and does not specify any information about the sign of the value or the scale. Therefore, scaling and sign must be applied to the read value according the parameter features [19].

For example, the 10^{th} parameter of the first group is the **Motor torque**. The value of this parameter is a signed number to within 0.1, which is in the range from -1600.0 to 1600.0. Thereby, after reading from the drive, the following sequence of procedures must be applied to the value:

– at first, a sign should be applied;

– then, the 16 bit value must be converted to an integer;

– finally, the integer value must be divided by 10.

3.4 MODBUS protocol

MODBUS is a messaging protocol of the application layer. It applies a master-slave principle of communication between devices connected on different types of buses or networks. MODBUS protocol allows an easy communication within all types of network architectures. It has a simple and elegant structure, this makes it popular for communication of automation devices. MODBUS can be accessed at a reserved system port 502 on the TCP/IP stack [20].

3.4.1 MODBUS messages

MODBUS is a request/reply protocol. The client device requests from a server services specified by function codes. The function code along with data block compose the Protocol Data Unit (PDU). PDU is independent of the lower communication layers and it can be supplemented with additional fields on the Applicational Data Unit (ADU). The visualisation of the MODBUS frame you can find in Figure 6 [20, 21].

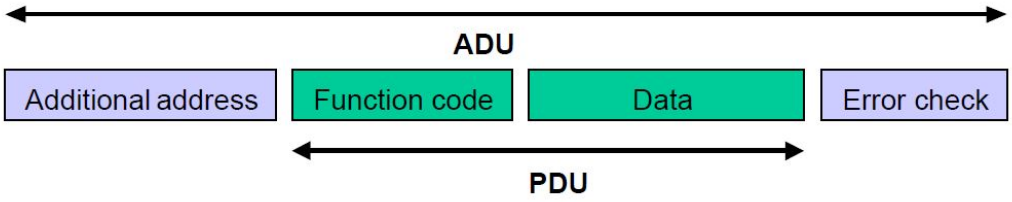


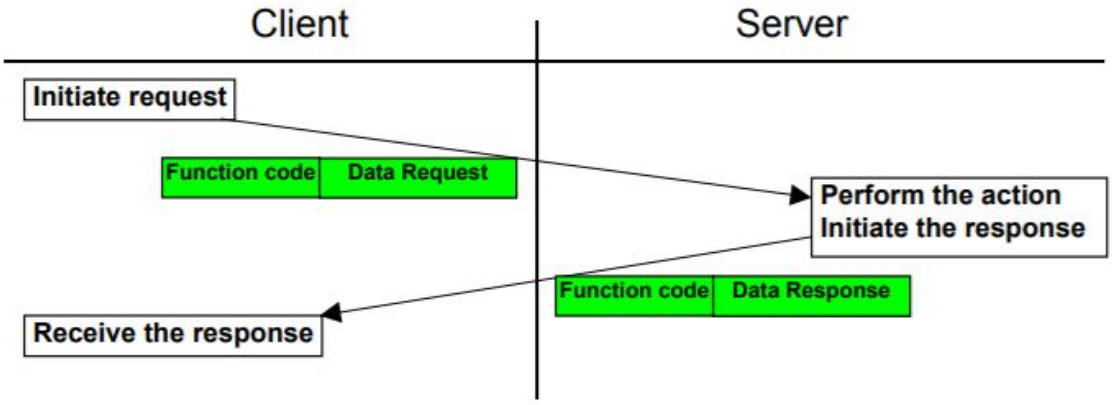
Figure 6: The MODBUS frame [21].

One byte of MODBUS Data Unit is reserved for the function code. The valid function codes are in the range of 1 – 255. The range of 128 – 255 is reserved for exceptional responses [20].

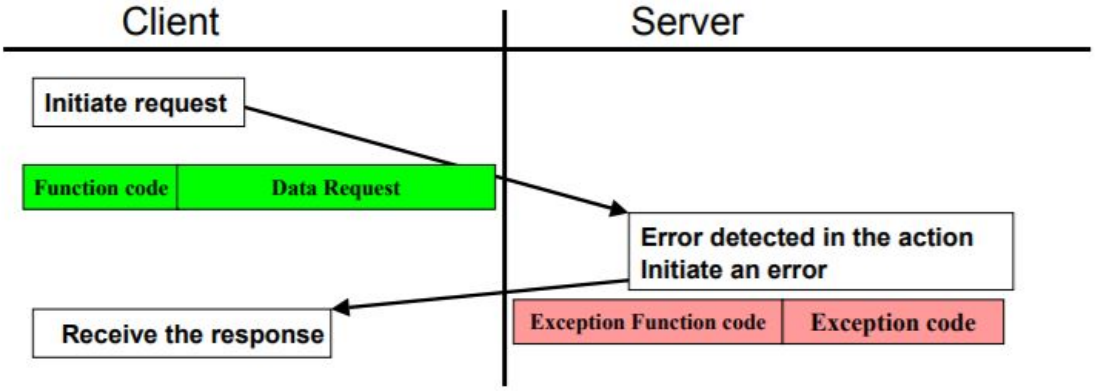
The data field of the MODBUS request message contains information, which is necessary for the server to perform the request. It can be register address, number of items to be handled, the count of the actual data bytes in the field [20].

3.4.2 MODBUS transactions

The function code of the error free response message is identical to the function code of the request message. The data field contains the response of the server. In case of error, the function code converts to the exception function code and data field contains an exception code [20]. The schemes of error free and exceptional response transactions can be found in Figure 7.



(a) The MODBUS error free transaction.



(b) The MODBUS exceptional response transaction.

Figure 7: The MODBUS transactions [20].

3.4.3 MODBUS data model

The MODBUS data model consists of a series of tables, which have different characteristics. Traditionally, the addresses of the table entities starts with a number, which specifies the table [20]. The four primary tables are described in Table 3.

Table 3: The MODBUS Primary Tables [20].

Primary tables	Object type	Type of	Address Numbers
Discretes Input	Single bit	Read-Only	1
Coils	Single bit	Read-Write	2
Input Registers	16-bit word	Read-Only	3
Holding Registers	16-bit word	Read-Write	4

3.4.4 MODBUS function code description

There are four function codes for data reading:

- 01 (0x01) Read Coils
- 02 (0x02) Read Discrete Inputs
- 03 (0x03) Read Holding Registers
- 04 (0x04) Read Input Registers

The request message in addition to the function code contains the starting address and the quantity of items to be read. The response message contains function code, byte count and requested status or value. In case of error, the message consists of error code and exception code, which can be 01 or 02 or 03 or 04 [20].

The MODBUS writing functions are divided into two groups: single item writing and multiple values writing. The functions have the following names and codes:

- 05 (0x05) Write Single Coil
- 06 (0x06) Write Single Register
- 15 (0x0F) Write Multiple Coils
- 16 (0x10) Write Multiple Registers
- 22 (0x16) Mask Write Register
- 23 (0x17) Read/Write Multiple registers

The request message of the functions for single item writing contains the function code, the address of the item and the value. The successful response is the copy of the request message [20].

The request message of the functions for multiple item writing contains the function code, the starting address, quantity of outputs, byte count and outputs value. The successful response is the function code, the starting address and the quantity of outputs [20].

The function code 22 (0x16) Mask Write Register is used to modify the contents of a holding register using a combination of an AND mask, an OR mask, and the register's current contents. [20]

The request message of this function contains the function code, reference address, And_Mask and Or_Mask. The response message is an echo of the request [20].

The function code 23 (0x17) Read/Write Multiple registers performs a combination of read and write operation in a single MODBUS transaction. [20]

The request message if this function consist of the function code, the starting address of the read items, the quantity of read items, the starting address of the write items, the quantity of the write items, the byte count of the write items and the value of the write registers. The response message contains the function code, the byte count and the value of the read registers [20].

In case of error, the response of the writing requests is similar to the response of the reading requests [20].

The rest of the MODBUS functions codes are:

- codes for diagnostic functions:
  - 07 (0x07) Read Exception Status (Serial Line only)
  - 08 (0x08) Diagnostics (Serial Line only)
  - 11 (0x0B) Get Comm Event Counter (Serial Line only)
  - 12 (0x0C) Get Comm Event Log (Serial Line only)
  - 17 (0x11) Report Slave ID (Serial Line only)
- codes for files and queue access:
  - 20 (0x14) Read File Record
  - 21 (0x15) Write File Record
  - 24 (0x18) Read FIFO Queue
- and the function code for tunnelling service requests and method invocations – 43 (0x2B) Encapsulated Interface Transport [20].

3.4.5 MODBUS exception codes

MODBUS protocol defines the following exceptional codes:

- 01 Illegal Function

- 02 Illegal Data Address
- 03 Illegal Data Value
- 04 Slave Device Failure
- 05 Acknowledge
- 06 Slave Device Busy
- 07 Negative Acknowledge
- 08 Memory Parity Error
- 0A Gateway Path Unavailable
- 0B Gateway Target Device Failed to Respond

More information about MODBUS functioning and detailed description of the MODBUS function codes and MODBUS Exception codes can be found in [20] and [21].

3.4.6 The Pymodbus library

The Pymodbus library is a fully implemented protocol stack in Python. The Pymodbus library documentation, which contains Pymodbus Library Examples and Pymodbys Library API Documentation can be found in [22].

In this work, the communication between a PC and an industrial drive has been performed with an application, which uses the Pymodbus library class **client**. This class allows to create a MODBUS client on the PC. The client can extract data from the drive and write parameters to the drive.

The code of the application, which has been developed for the thesis, can be found in Appendix F.

4 Data collecting

4.1 Chipset producer questionnaire

ARF requires a set of computers and radio transceivers, typically not applicable to the industry. At the time this thesis was in process, there was no mass production of NB-IoT chipsets. Only few press releases of some companies about NB-IoT chipset development could be found in Web. Therefore, the first part of the Master's thesis work in a chronological order was to survey NB-IoT chipset producers and gather information about NB-IoT modules (ABB is interested the most in USB modules): their release time, price, value chain and feature set. For this part of the thesis a web questionnaire was used.

4.1.1 The NB-IoT chipset vendor questionnaire form

Since the press release papers do not give much information about the products, a questionnaire was selected to be created in this thesis. The system, which has been used to make the survey is webropolsurveys.com, where Aalto users have a permission to create online surveys. The questionnaire is presented in Appendix A. The following information was collected:

1. Name of the company (it was used for filtering purposes);
2. Time (estimate) when the chipsets will be available for industry/sale;
3. The estimated price of an NB-IoT module for a 1000 pieces bulk purchase;
4. The value chain in the NB-IoT equipment production:
   - Horizontal integration: chipset manufactures, module producers and module integrators are independent;
   - Vertical integration: chipset manufactures produce also final products (sensors).
5. How eSim support would suit the for NB-IoT modules?
6. What kinds of additional features the NB-IoT chipset will have?
7. Any additional comments by a responder.

4.2 Use cases of NB-IoT technology for industrial drive data transmission

The industrial processes are complex and large-scale. Therefore, advanced monitoring technologies are needed to provide efficient performance and safety of production. In this work, the application of the NB-IoT technology for industrial process monitoring is analysed.

The industrial processes are diversified and typically highly tailored for a specific production. It is a complex task to study various production processes and the application of the NB-IoT technology for them. Therefore, in this work, the peculiar NB-IoT features are considered at first. Afterwards, the environment suitable for the NB-IoT features is described.

The considerable features of the NB-IoT technology are:

- advantages of cellular technology and licensed band;
- the massive device support;
- the extended battery lifetime;
- the coverage extension.

The scenarios of production processes for the mentioned NB-IoT features are described in Section 6.4.

5 Measurement systems

The literature search and the survey of chipset providers show that the NB-IoT technology could be used for the transmission of the industrial drive data. However, as there was no NB-IoT chipsets available while the study was made, the performance of the technology had to be checked with the existing ARF. For this thesis, two testbeds have been developed with the purpose to study the NB-IoT technology for industrial process monitoring. The first testbed configuration is used for data transmission delay measurements from the NETA-21 monitoring tool to the NB-IoT base station via the ARF. The measurement of the transmission delay helps to study the applicability of NB-IoT technology for industrial process monitoring via the NETA-21 monitoring tool. The second set is a demo set developed to validate the ARF performance for industrial drive monitoring. The set provides possibility to transmit the parameters from the industrial drive to the NB-IoT eNodeB and represent the received data by a web interface. The clarification scheme of the thesis procedure can be found on Figure 8.

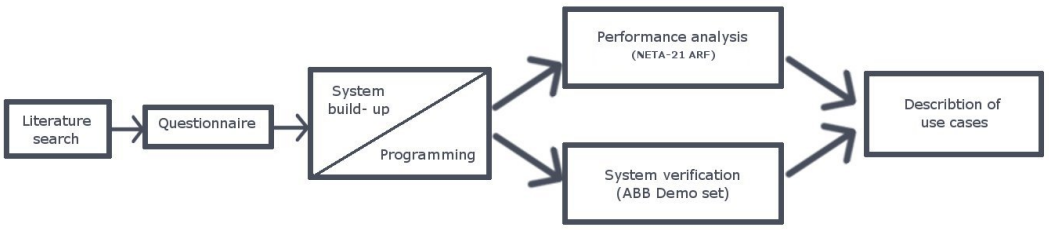


Figure 8: The clarification scheme of the thesis procedure. (Courtesy T. Kanerva)

5.1 The ARF with a NETA-21 monitoring tool

In this work, we are interested how NB-IoT can be used for collecting monitoring data from industrial drive via the NETA-21 monitoring tool. The general scheme of the communication plan can be found in Figure 9.

The NETA-21 monitoring tool collects data from the industrial drives and creates an XML file, which is transmitted to a cloud for further analysing. Average size of a packet, which NETA-21 receives from one industrial drive is 100 bytes. The maximum number of drives connected to NETA-21 is 10. Therefore, the size of the XML file varies from 100 bytes to 1000 bytes for a single data aggregation.

Typically, NETA-21 aggregates the industrial drive data during some period (for example, one minute) and then sends the collected data to the cloud. This aggregation data could be compressed. The example from [17] shows that the compressed file for 1 minute data aggregation from 9 drives is 55 kB, the uncompressed file is 914 kB [17]. Therefore, in such conditions, the data transmission requirements for compressed data is ≈ 1 kB/s and for uncompressed data is ≈ 15 kB/s.

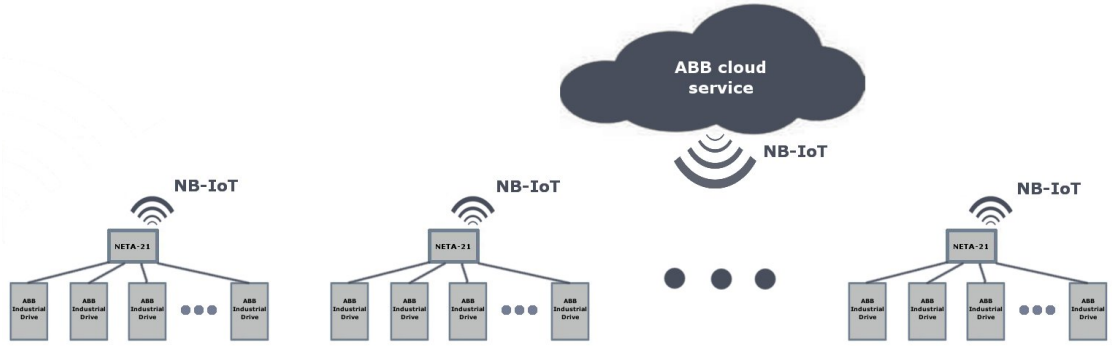


Figure 9: The NB-IoT transmission by NETA-21 monitoring tool. (Courtesy T. Kanerva)

To perform the transmission of the industrial drive data with an NB-IoT system, NETA-21 should have a NB-IoT chipset. This will require changes to the production process. For a pilot implementation of NB-IoT transmission, an external USB NB-IoT module could be used. Since the external USB NB-IoT modules are not available at the moment, the initial test of the performance of NB-IoT technology for data transmission from the NETA-21 monitoring tool can be done by ARF.

To perform the data transmission from the NETA-21 by ARF, a test set has to be developed. In this Master's thesis, I will call this set *NETA-21 ARF*. NETA-21 ARF consist of hardware and software products. The set performs measurements of transmission delay of data packet, which have been generated on NETA-21 monitoring tool and then transferred to the NB-IoT eNodeB through ARF.

5.1.1 The implementation of the NETA-21 ARF set

ARF is a technically demanding software. It requires a high performance by the computer processor. In turn, the monitoring tool NETA-21 was not designed for such demanding software. Therefore, the system as such can not run the NB-IoT software.

To perform transmission delay measurements with the developed system, the data from the NETA-21 monitoring tool should be firstly sent to a computer, which runs ARF.

As the monitoring process is mostly sensitive to the uplink delay, the NETA-21 ARF set was configured so that it measures the uplink transmission delay from the NETA-21 monitoring tool to the NB-IoT eNodeB. The main problem of measuring of one way transmission delay is clock difference between transmitter and receiver. In other words, the components of the measurement set must be synchronized so their timestamps would be comparable. Synchronization for NETA-21 ARF is difficult to perform, therefore, the measurements of the transmission delay were split to two parts.

At first, a round trip delay between NETA-21 and the computer, which rans ARF, was measured. The connection between NETA-21 and the computer is a typical Ethernet network so the uplink and downlink delay is approximately the same and the uplink delay would consist of a half round trip delay.

The second part is measuring the uplink transmission delay for ARF and this was performed so that the UE side of ARF and eNodeB side were ran on the same computer. Thereby, the transmitter timestamps and the receiver timestamps are the compatible.

The scheme of the communication process could bee seen on Figure 10.

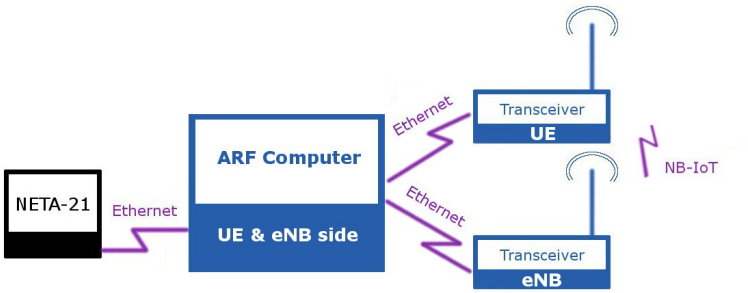


Figure 10: Configuration of NETA-21 ARF set. (Courtesy T. Kanerva)

The computer, which runs ARF, can be a typical desktop/laptop or it can be an SBC. In this work two sets of measurements were performed: the first NETA-21 ARF set was ran by a typical laptop with Quad-Core Central Processing Unit (CPU), 8Gb of Random Access Memory (RAM) and Linux operation system, and the second set was ran by an SBC.

ODROID-C2 is the SBC, which has been used for running the second measurement set. This computer has been chosen because of the following reasons:

- it has small size, which allows to make the whole system smaller in physical size;
- it has a powerful processor enough fast to run the programs;
- it has a real Ethernet connector without a USB bridge, making it possible to perform reliable and fast transmission between the sensor and the USPR.

Thereby, it could be checked whether the usage of SBC (which is smaller and more convenient for this purposes) instead of typical computer brings some limitations or not.

The control of the NETA-21 monitoring tool can not be performed directly because NETA-21 has not a port for direct output. Yet NETA-21 has a Linux-operating system and, therefore, it can be controlled with SSH. In this thesis, the same computer, which runs the ARF, was used to control the NETA-21 monitoring tool with SSH.

5.1.2 The NETA-21 ARF hardware components

The hardware components of NETA-21 ARF set are presented on Figure 11 [12, 23] with the division of new hardware components and the existing hardware components.

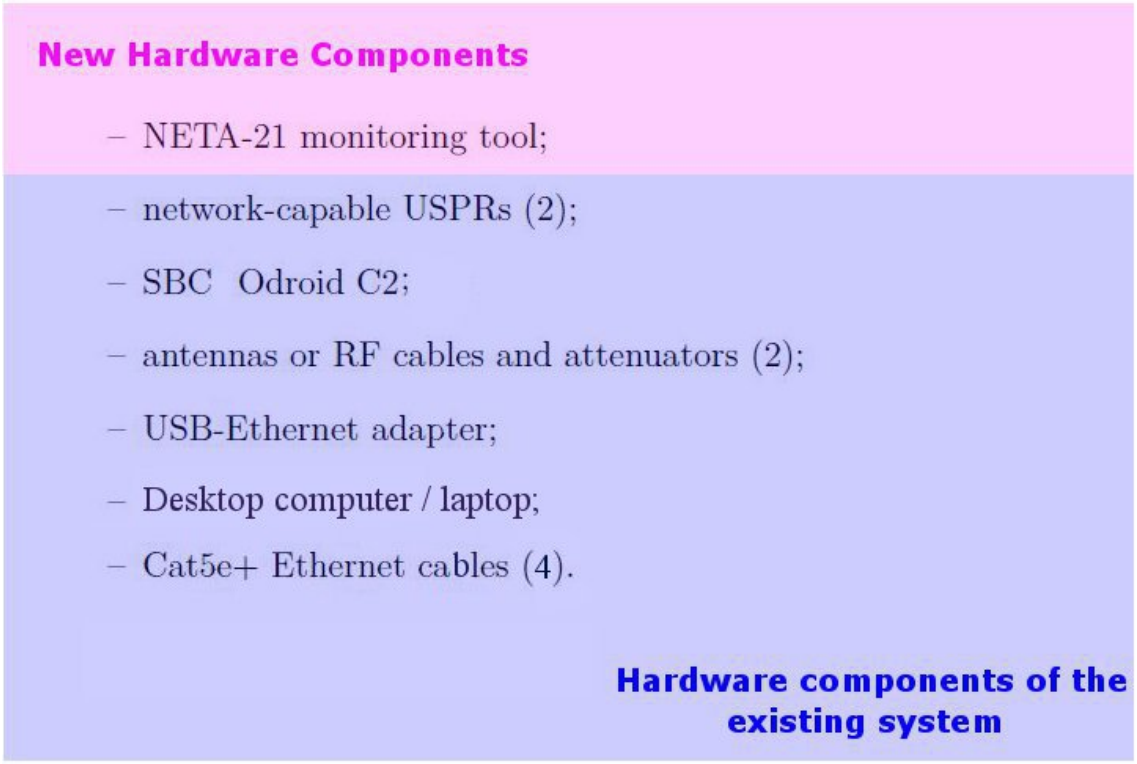


Figure 11: The Hardware components of NETA-21 ARF set. The pink color refers to the new hardware components. The blue color refers to hardware components of the existing ARF system. (Courtesy T. Kanerva)

5.1.3 The NETA-21 ARF software components

NETA-21 ARF set software components are presented on Figure 12 with the division of new and modified software components.

5.1.4 The performance measurements

To check how well ARF meets the requirements for industrial drive data transmission and whether ARF brings some limitations to industrial drive monitoring process, a delay of packet transmission from the NETA-21 monitoring tool to eNodeB should be measured.

As NETA-21 cannot run the ARF, the evaluation of data transmission delay from NETA-21 to NB-IoT eNodeB should be divided to two separate measurements. The first measurement evaluates a round trip delay of data transmission between NETA-21 monitoring tool and a computer, which runs the ARF. The second measurement estimates uplink transmission delay for ARF.

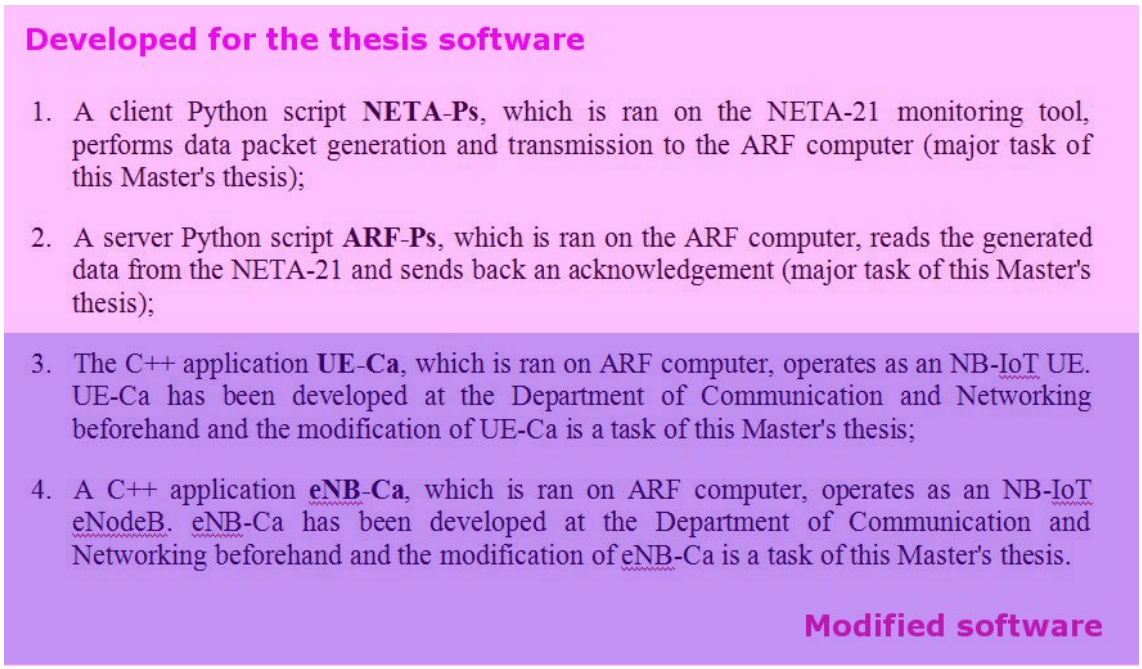


Figure 12: The Software components of NETA-21 ARF set. The pink color refers to the software developed for this thesis work. The purple color refers to the existing software components, which have been modified in this Master's thesis. (Courtesy T. Kanerva)

5.1.4.1 NETA-21 – ARF computer round trip delay measurement

The software for round trip delay measurement between NETA-21 and ARF computer is a client-server application based on network sockets. The software consists of two components:

- the client script – NETA-Ps, which is ran on the NTA-21, performs data packet generation, transmission of the packet to the ARF computer and calculating of round trip delay after receiving the acknowledgement from the ARF computer;
- the server script – ARF-Ps, which is ran on the ARF computer, reads the packet from the NETA-21 and sends an acknowledgement back to NETA-21.

The software components are written using Python language. The Python language was chosen because the NETA-21 monitoring tool contains a Python interpretor and all the other languages would require installation of a compiler or a cross-platform compilation. However, the NETA-21 does not have enough memory space for the installation of an additional compiler, and cross-platform compilation takes a lot of time for transmission of a binary file to the monitoring tool.

The scripts have the features listed in the following.

1. NETA-Ps – the client side (NETA-21 monitoring tool) performs:
   - generation of data packet of a specified size (NETA-21);
   - sending of a data packet to a server side (UE SBC);
   - receiving an acknowledgement from the server;
   - measuring of the round trip delay;
   - waiting for a specified time interval.
2. ARF-Ps – the server side (ARF computer) performs:
   - receiving of data from NETA-21 monitoring tool;
   - sending an acknowledgement back to NETA-21 monitoring tool;

The NETA-Ps (the client side) generates a data packet of a specified size and sends it to the ARF computer, where ARF-Ps (the server side) is ran. The ARF-Ps receives the packet and sends an acknowledgement (UE ack) that UE has got the packet from the NETA-Ps. The NETA-Ps receives the acknowledgement and calculates the NETA-21 – ARF computer round trip time.

NETA-Ps reads the following input parameters from a configuration file:

- packet size, adjusted by setting a number of drives connected to NETA-21;
- time interval, set in seconds;
- the number of iterations (defines how many times the NETA-21 monitoring tool should send data to eNodeB and receive an acknowledgement);
- IP address of a server (IP address of UE SBC);
- Port number, which the server is listening.

The code of the application can be found in Appendix F.

5.1.4.2 ARF uplink transmission delay measurements

ARF uplink transmission measurements were made using a single computer, which runs both UE and eNodeB side of the ARF, so the one way transmission delay can be calculated. At first, the measurements were performed on a typical computer and then on an SBC to compare performance of the system ran by different types of computer.

The measurements were performed so that a timestamp was taken when UE side was about to transmit a packet to the eNodeB side. At the eNodeB side, a timestamp was taken after the receiving of the packet. Hereby, the difference of these timestamps shows the uplink transmission delay. The results of the measurements can be found in Section 6.2.

The scheme of the NETA-21 ARF operation is shown on Figure 13. Further development of ARF would allow to make better and more careful measurements of uplink transmission delay, downlink transmission delay and round trip delay of NB-IoT data exchange.

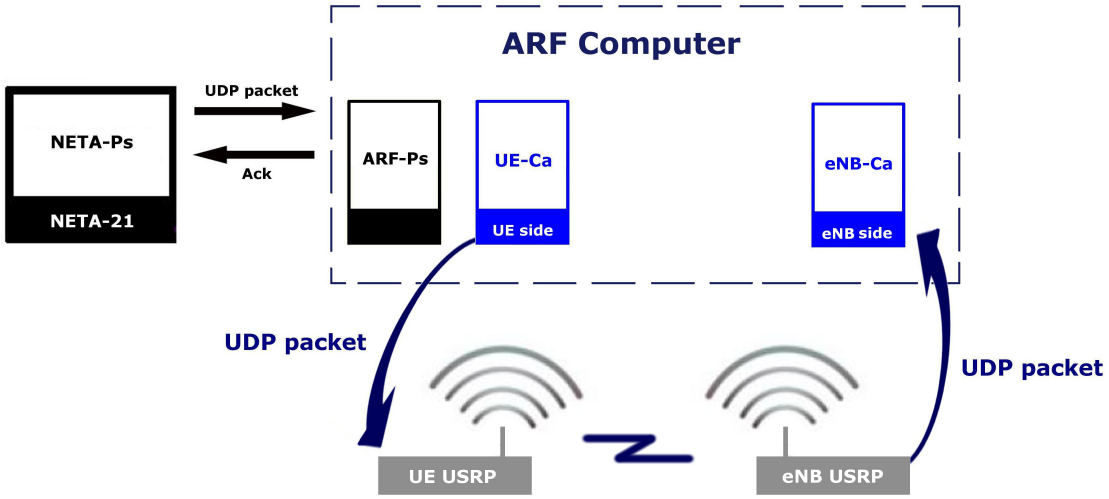


Figure 13: Illustration of the NETA-21 ARF operation. (Courtesy T. Kanerva)

5.2 The system validation: ABB demo set

To validate the ARF system for industrial drive data transmission a demo set is developed. The demo set is based on the existing NB-IoT demo where the parameters of a meteorological sensor are read, sent with ARF to eNB and presented with a web interface. To apply the system for data transmission of ABB industrial drive, the UE-Ps script, which reads the industrial drive parameters has been developed and the representation module of ARF has been modified. Finally, the ABB demo set has been presented on the conference EuCNC 2017 in Oulu 12.06.2017. In addition, a poster has been developed for the conference. It can be found in Appendix B.

In the ABB demo set, the parameters of an industrial drive are read via a Modbus/TCP gateway and then transmitted with ARF to an eNodeB node, where the parameters are represented with a web interface.

To make the demo set smaller in size, the monitors and keyboards are not connected to the SBCs. Instead of it, the SBCs are controlled by laptops with SSH.

5.2.1 Hardware components

The hardware components of the demo set are presented on Figure 14 [12, 23] with the division of new hardware components and the existing hardware components.

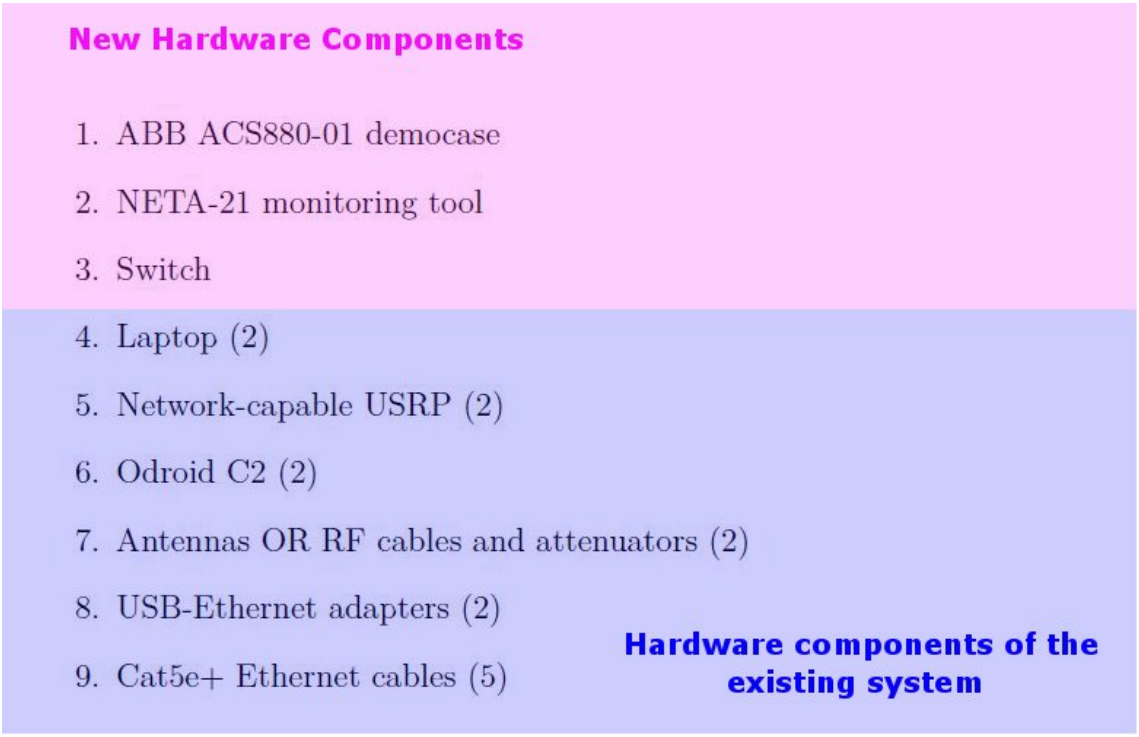


Figure 14: The Hardware Components of the ABB Demo set. The pink color refers to the new hardware components. The blue color refers to hardware components of the existing ARF system. (Courtesy T. Kanerva)

5.2.2 Software components

The software components of the demo are presented on Figure 15 with the division of new, existing and modified software components.

5.2.3 Principles of operation

The system consists of five separate parts: the first one, which is represented by the UE-Ps, reads data from the industrial drive (in the demo it is not a real drive but an ABB democase). The reading is carried out by using MODBUS protocol through the NETA-21 monitoring tool, which plays a role of a TCP/Modbus gateway. The read data is then sent to the TUN interface where it is taken by the UE-Ca. The Ue-Ca is the second part of the system. The UE-Ca NB-IoT application then transmits the packet with the NB-IoT technology to the third part – the eNB-Ca. The eNB-Ca sends the data to a localhost, where the fourth part – the eNB-Ps – listens for it. The eNB-Ps, in addition, provides an HTTP server, to which it is possible to connect with a browser. The last part of the system is a Javascript program, which regularly fetches new values and updates the display using them. The scheme of the system can be found in Figure 16.

In the Results section, you can find more detailed description of reading data from the industrial drive and presenting the data with the web interface. These parts of the system have been developed and modified in this thesis.

Developed for the thesis software

1. A Python script **UE-Ps**, which reads the industrial drive parameters (major task of this Master's thesis);
2. HTTP-server Python script **eNB-Ps**, which receives data on the eNodeB side. eNB-Ps has been developed at the Department of Communication and Networking beforehand and the modification of eNB-Ps is a task of this Master's thesis;
3. Javascript script **eNB-Js**, which represents the data on the eNodeB side. eNB-Js has been developed at the Department of Communication and Networking beforehand and the modification of eNB-Js is a task of this Master's thesis;
4. A C++ application **UE-Ca**, which is ran on the UR SBC, operates as an NB-IoT UE. UE-Ca has been developed at the Department of Communication and Networking beforehand;
5. A C++ application **eNB-Ca**, which is ran on the eNodeB SBC, operates as an NB-IoT eNodeB. eNB-Ca has been developed at the Department of Communication and Networking beforehand.

Existing software

Figure 15: The Software Components of the ABB Demo set. The pink color refers to the software developed for this thesis work. The blue color refers to the existing software components, which have been developed earlier in the Department of Communications and Networking. The purple color refers to the existing software components, which have been modified in this Master's thesis. (Courtesy T. Kanerva)

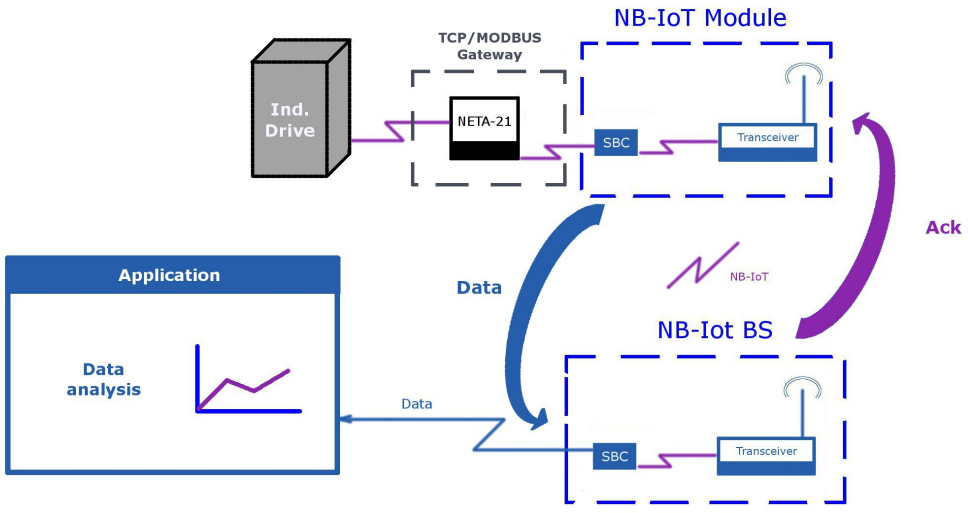


Figure 16: The scheme of the ABB demo set. (Courtesy T. Kanerva)

6 Results

This section presents the results of the subtasks implementation. Table 4 clarifies the work, which has been performed for achieving the research targets.

Table 4: The performed work for the subtasks of the Master's thesis.

The Subtask	**The Subtask Implementation**	**The work done**	**The sections**
The background check	The literature survey	Extraction of the relevant information from the found literature	3
Gathering of the information about NB-IoT chipsets	Creation of a chipset producer questionnaire	Searching for NB-IoT chipset producers and their survey	4.1, 6.1
Measuring of the performance of the ARF for the NETA-21 data transmission	Development of the NETA-21 ARF set	Adaptation of ARF for the NETA-21 data transmission: development of NETA-Ps and UE-Ps, modification of the eNB-Ps	5.1, 6.2
Verification of the operation of ARF for the data transmission of industrial drives	Development of ABB demo set, which verifies the system operation	The adaptation of ARF for the data transmission of industrial drives: development of UE-Ps and the modification of eNB-Ps and eNB-Js	5.2, 6.3
Determination of scenarios for the NB-IoT system usage while monitoring the industrial drives	Creation of possible use cases for industrial drives monitoring via NB-IoT	Reporting of examples of possible use cases for industrial drives monitoring via NB-IoT	4.2, 6.4

6.1 The chipset producer questionnaire

The implementation of the NB-IoT technology with the Software Radio is infancy and not yet in level where it can be integrated into mission critical industrial processes. Hereby, Particular implementation of NB-IoT based on ARF platform can be used to analyse the applicability of the NB-IoT for the monitoring of the industrial drives. To apply the technology for the industry, the NB-IoT modules or chipsets are required. The NB-IoT chipset application will lead to modifications in the industrial drive production. It will be a lengthy process. Therefore, at the beginning, the NB-IoT USB modules can be used to apply the NB-IoT technology for the monitoring of the industrial drive.

The NB-IoT chipsets are not available yet, therefore, the gathering of the information about the NB-IoT chipsets is an important tasks for this Master's thesis.

6.1.1 The chipset vendors

At first, the websites of the chipset vendors have been checked and the press releases of the NB-IoT chipsets have been analysed. After the web search, the following list of NB-IoT chipset producers was composed:

1. Quectel Wireless Solutions
2. Altair (a Sony company)
3. Intel
4. Qualcomm
5. Sequans Communications
6. U-Blox
7. Nordic Semiconductor
8. Spreadtrum Comm.
9. Huawei
10. MediaTek
11. Telit

A link to the created survey has been sent to all the mentioned companies with a request to answer the questions. Two companies have responded to the request. They are:

1. Quectel Wireless Solutions
2. Altair (a Sony company)

As there are only two respondents the answers were processed manually based on automatic generated report in webropolsurvey.com system.

6.1.2 Questionnaire results

Based on the answers of the online questionnaire, the following information was provided:

6.1.2.1 Avaliability of the NB-IoT chipsets

For the question about the avaliability of the NB-IoT chipsets, the Quectel company answered, that the chipsets will be available in several months. In turn, the Altair company promised to start selling of the NB-IoT chipsets in the beginning of 2018 year.

6.1.2.2 Price of the NB-IoT module

The both companies specified the same price range of the NB-IoT chipset for a 1000 pieces bulk purchase – 7–10 $. This price is higher than is generally specified price of a single module, which is 5 $. The higher price of the module could be explained by the novelty of the technology. The costs, possibly, could decrease over time.

6.1.2.3 Value chain of the NB-IoT chipset production

The Quectel company as well as Altair company defined the value chain of the NB-IoT chipset production as a horizontal chain. This means that the chipset manufactures, the NB-IoT module producers and the module integrators will be independent company according the questioned companies opinion.

6.1.2.4 eSim support

The Quectel company answered for this question that the eSim will be integrated into their chipsets. Unfortunately, the Altair company could provide the information about eSim support only under NDA.

6.1.2.5 The feature set of the NB-IoT chipset

The Quectel chipsets besides the specified in the 3GPP Release 13 features will have the GPS/GNSS support, the application processors integration and the application development platform. The Altair company could provide information only about the supported application development platform. Other additional features of the Altair NB-IoT chipsets could be provided only under NDA.

6.2 NETA-21 ARF test set

As there is no available NB-IoT chipsets or modules, the operation of the NB-IoT technology for monitoring of the industrial drives is checked with the NB-IoT implementation on ARF platform, which has been developed at the Department of Communications and Networking, Aalto University.

The collection of the information about the industrial drives is performed by the NETA-21 monitoring tool. The NB-IoT system can perform transmission of the collected data from the NETA-21 monitoring tool to the cloud.

To check the performance of the NB-IoT system, a testbed with NETA-21 connection to NB-IoT UE implementation and from there to NB-IoT BS has been created. The testbed allows to transmit randomly generated data from the NETA-21 monitoring tool to the NB-IoT eNodeB and measure the transmission time.

6.2.1 Data transfer results

The data transmission delay measurement between the NETA-2 monitoring tool and NB-IoT eNB consists of two parts. The first part is NETA-21 – SBC transmission delay and the second part is ARF transmission delay measurement.

6.2.1.1 NETA-21 – SBC transmission delay

Data transfer between NETA-21 monitoring tool and UE SBC was observed to be stable because there is no data packet corruption or lost and the transmission delay remains constant. The round trip time of a data packet from NETA-21 to SBC and back (without sending data to eNodeB) is 1–3 ms. Data packet size, which varied from 100 bytes to 1000 bytes, did not affect significantly the round trip time.

The reason for the significant delay (1–3 ms) of the transmission between NETA-21 and UE SBC could be the low performance parameters of the NETA-21 and the slow data transfer through the Ethernet-USB adaptor.

6.2.1.2 ARF transmission delay

ARF transmission delay has been measured with two testbeds. The first testbed was ran by a typical computer. The second testbed was ran by an SBC. The summoned results of the measurements are presented in the figures below.

First, in Figure 17 is shown the delay of ARF, which is ran by a typical laptop. The measurements were done for 4, 8, 16, 32 and 128 repetition factor. The repetition factor represents the number of repetitions. In turn, high amount of repetitions are used to increase probability of successful decoding of a corrupted data packet although it increases the transmission delay.

As shown by the plot, the increase in the transmission delay is clearly monotonic with the repetition factor. However, it is difficult to describe the behaviour of the growth of delay, i.e. whether it is linear or stepwise or exponential as a function of number of repetitions. What is very important to note, the standard deviation increases very fast with a growing number of repetitions. In detail, if for 4–16 repetitions covariation is $\approx 25\%$ (average) then for 128 repetition factor it is $\approx 40\%$. The magnification of the plot shows that the absolute deviation is essentially constant at low numbers of repetitions.

Second, in Figure 18 is shown the delay of ARF ran by an SBC. SBC has typically lower performance parameters than a laptop computer.

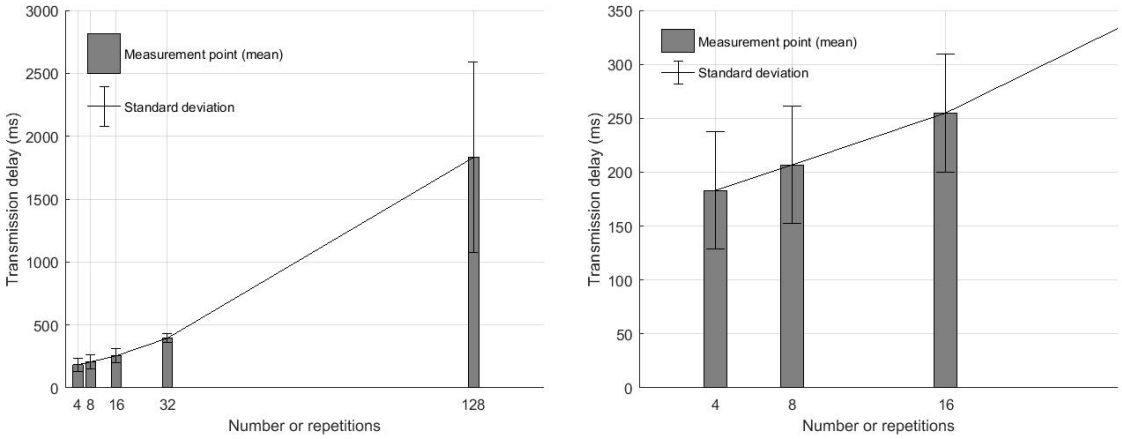


Figure 17: The plot showing measurements of ARF transmission delay performed on a standard computer. Magnification of the plot is on the right.

Indeed, the graph shows that the measured value of the delay for a low repetition factor is slightly higher for the system implemented on an SBC than for the system ran by a typical computer. Most probably, the reason for the longer delay is the lower performance parameters of the SBC. Interestingly and unexpectedly, the delay of the transmission with a repetition factor of 128 is lower and more stable for the system ran by the SBC than for the system ran by a typical computer. In terms of the standard deviation of measured transmission delay, the increase in the deviation is slower than the mean delay value. Therefore, although it can be seen that the standard deviation value grows with increasing repetition factor, the covariation decreases from 30% at a repetition factor of 4 to 4% at a repetition factor of 128. The magnification of the plot shows that the absolute deviation is essentially constant throughout the measurement range. The reason for such an unstable transmission performed by the laptop computer could be the unintended behaviour of USRPs or side software running on the computer, which execution could affect performance of the system.

To conclude for both systems, the average delay for a low repetition factor, which is typical for a good network coverage condition in reality, is approximately 200 ms. For a higher repetition factor (say 128 repetitions, simulating worse coverage conditions), the transmission delay value is more than one second.

Another interesting topic to analyse is the comparison of time, which is spent for actual data transmission, and the time, which is used by the equipment to process the tasks of data transmission. In the NB-IoT implementation of ARF, resource unit number is set to 6. It means that average data transmission interval with minimum delay of repetitions can be calculated as $number\_of\_repetitions \times 6$ ms. Hereby, it can be seen that the processing time is considerably greater then the transmission time. I.e. the slop of both curves is greater than 6 ms per repetition. This implies that usage of equipment with better performance parameters could improve work of the system and decrease transmission delay.

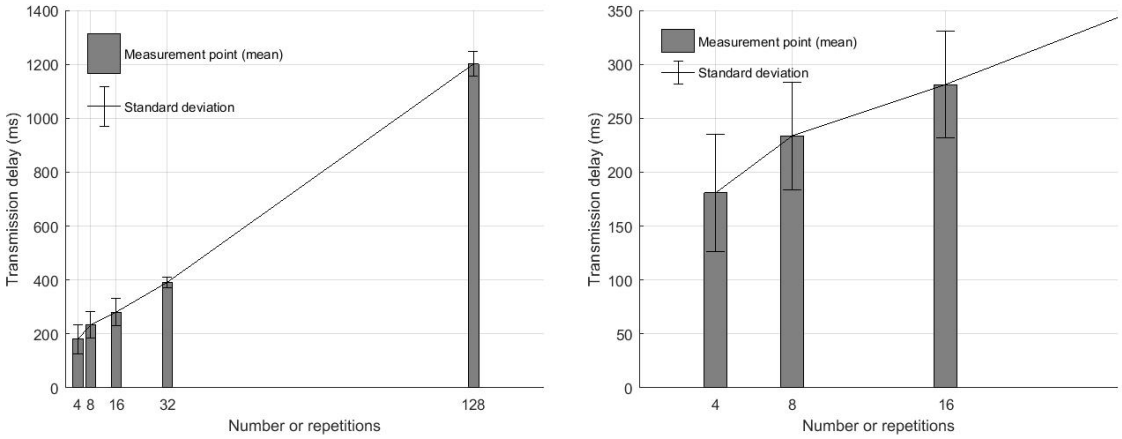


Figure 18: The plot showing measurements of ARF transmission delay performed on an SBC. Magnification of the plot is on the right.

6.2.1.3 Transmission requirements for ARF

Since the transmission delay between the NETA-21 monitoring tool and an SBC is small (1–3 ms) compared to the transmission delay of the ARF (≈200–1000 ms), it could be neglected. Therefore, for a low (4–16) repetition factor, the rate of 1000 bytes packet transmission is ≈5 kB/s. For a repetition parameter 128, the transmission speed of 1000 bytes packet is less than 1 kB/s.

If one considers a basic example from [17], the requirements for the transmission of data, which have been aggregated from nine drives during one minute, are: 15 kB/s for uncompressed data and 1 kB/s for compressed data. Therefore, in this case, the usage of ARF for industrial drive data transmission is possible if the data are compressed and transmission conditions are good enough. Decreasing the data aggregation interval or decreasing the number of supported drives can help the transmission to fit the limitations, which ARF brings.

6.3 The system validation: ABB demo set

To validate the ARF performance for the data transmission of the industrial drives, a demo set has been created. The demo set performs reading of several parameters from an industrial drive via NETA-21 MODBUS/TCP Gateway. After reading, the parameters are transferred with ARF to the NB-IoT eNodeB and represented on the eNodeB side by a web interface.

6.3.1 The starting of the system

Before running the applications, the SBCs must be configured properly: TUN interface should be set and IP addresses assigned [12]. The detailed instructions for the SBC configuration can be found in Appendix D.

After the initial settings, the system can be started. The order to run the applications is not important: the client sides can send requests to a port, which is

not listened, and the server sides can completely stay in waiting mode.

6.3.2 Reading of data from the ABB industrial drive

The reading of parameters from the ABB industrial drive is performed by a Python script, which is ran on UE SBC – UE-Ps. The script reads the parameters with a MODBUS protocol through NETA-21, which works as a Modbus/TCP gateway. After reading the parameters, the script processes them, concatenates, creates a UDP packet of this data and sends the packet to the configured TUN interface. At the interface the data is taken by NB-IoT system for further transmission.

The reading of the parameters is performed sequentially, one by one. Sometimes, the drive is busy and cannot answer for the request. To avoid this, an error recover procedure was introduced into the demo system: the connection is re-established and the next request is sent only after 5 seconds. If the drive is still busy and cannot respond, the reconnection procedure repeats.

6.3.2.1 The configuration of NETA-21 monitoring tool

The first step in organizing the reading of industrial drive parameters is the activation of Modbus/TCP gateway function of the NETA-21 monitoring tool. The instruction of the function activation can be found in Appendix D. After the NETA-21 monitoring tool is configured, it automatically processes Modbus requests from the UE SBC, which is connected to the NETA-21 via Ethernet.

6.3.2.2 The ABB democase as an industrial drive

In the demo set, the industrial drive is represented by an ABB ACS880-01 democase. The democase includes a drive ACS880, a small motor, a mechanical brake, a standard industrial drive control panel, and an additional control panel, which contains easy-to-use buttons and knobs. The both control panels are placed on the same board and connected to the drive through a panel bus port. In the following text, a term *input control panel* for the standard control panel of the drive and *remote control panel* for the additional control panel are used.

The feature set of the democase is limited in comparison to a real drive. The issue is described in detail in Section 6.3.4.

6.3.2.3 The connection of the ABB democase to the NETA-21 monitoring tool

The democase can be turned on only when the control panel board is connected to the drive. The reason for this is that the control panels are powered by a Panel bus connection and the democase detects a *Power failure* without the control panels being set on. To use the NETA-21 monitoring tool as a Modbus/TCP gateway, it must be connected to the industrial drive with the same Panel bus connection. Since there was no Panel bus switch/hub, the same port has to shared between the control

panel board and NETA-21 monitoring tool. There is the following procedure to share the port:

- first, the **I/O** output of the industrial drive must be connected to the control panel board;
- second, the drive could be turned on and motor spin could be initiated;
- third, the Ethernet cable must be taken from the control panel board and connected to the first Panel bus port of the NETA-21 monitoring tool.

It should be noted that the cable must be connected to the first Panel bus port of the NETA-21 monitoring tool because the addressing of drives connected to NETA-21 depends on the port it is connected to. There is no need for the demo set to make configurable drive addressing because there is only one drive and one monitoring tool. Therefore, the address of the drive is fixed in the UE-Ps script. If the democase is connected to the second port, the UE-Ps script can not find the drive and is not able to read the parameters.

6.3.2.4 The parameters of the industrial drive

To validate the system behavior, four different industrial drive parameters have been selected. All the parameters belong to the first parameter group *"Actual values"*. Therefore, their address starts with a digit **1**. The parameters are [19]:

- motor speed (parameter address 100);
- output frequency (parameter address 105);
- motor current (parameter address 106);
- motor torque (parameter address 109).

The parameters are read with the MODBUS/TCP gateway of the NETA-21 MODBUS function code 03 *Read Holding Registers*. In firmware 2.23, the settings of the MODBUS/TCP gateway could be configured to support also a read/write option. To know more about the writing of parameters, you can read Section 6.3.4.

6.3.2.5 The processing of the industrial drive parameters

By a default, MODBUS protocol returns after reading from an industrial drive the internal raw 16-bit value and does not specify any information about the sign and scaling. Therefore, the following format has been applied for the specified parameters [19]:

- motor speed is a signed number to within 0.01 in the range -30000.00 – 30000.00 rpm;

– output frequency is a signed number to within 0.01 in the rage -3000.00 – 3000.00 Hz;

– motor current is a positive number to within 0.01 in the range 0.00 – 30000.00 A;

– motor torque is a signed number to within 0.1 in the range -1600.0 – 1600.0 %.

The motor speed and frequency are signed numbers because the spin of the motor can be clockwise (positive) or anticlockwise (negative) [19].

Thr formatting is performed on eNB side because, after the reading of the values, the parameters are transformed to chars (unsigned byte values), concatenated to a string and sent with a UDP packet to the TUN interface. At TUN interface, the NB-IoT system accepts them and transmits to NB-IoT eNB.

6.3.3 The data representation module

After the parameters are transmitted through the NB-IoT system they are sent, to the eNB-Ps script. The script accepts the data, splits the parameters, stores them as separate values and applies the formatting mentioned in the previous section.

The Python script also runs an HTTP server to which user must connect. Inside the Index.html file a Javascript – eNB-Js program fetches values and outputs them on the display [12].

The page consists of four blocks for representing the current values of the industrial drive parameters and their average.

6.3.4 The two-way communication

Theoretically, MODBUS protocol can be used for writing parameters to an industrial drive. The ability to write parameters allows to optimize the performance of the drive remotely. For the developed demo case, usage of MODBUS writing function would demonstrate the two-way communication between the UE represented by an industrial drive and the monitoring application, which is ran on the eNodeB. However, writing of parameters to the industrial drive via the NETA-21 monitoring tool playing role of a MODBUS/TCP gateway is possible only through an XD2D connector. The connector presents on a real drive but it is absent on the ABB democase. This makes the implementation of the two-way communication for the demo set impossible.

In case of real drive instead of the democase, the following procedure must be apply to perform writing of parameters to the industrial drive [14, 24].

1. At first, MODBUS network settings have to be configured: for the Fieldbus control through the embedded fieldbus interface (EFB) group 58 must be configured:
   – the parameter 58.01 must have a state *enable*;
   – the correct baud rate and parity (the same as in NETA-21 settings) must be set.

2. The drive must be configured to allow control from MODBUS:
   - the parameter *20.01 Ext1 commands* must have the subfield *Embedded fieldbus* enabled: this makes the *start* and *stop* commands to be taken from the Embedded fieldbus;
   - the parameter *22.11 Speed ref1 selection* must have *Other* subfield enabled.
3. If the Fieldbus control through a fieldbus adapter is used then groups to configure are 50 and 51 and in groups 20 and 22 instead of *Embedded fieldbus* subfield and *Other* subfield must be enabled subfields *Fieldbus A* and *FB A ref1* respectively.

After this configuration, the drive could be controlled using ABB Drives control profile. For more information see [14].

6.4 The use cases of NB-IoT technology for the industrial process monitoring

The NB-IoT technology has unique features, which can optimize the industrial process monitoring for some specific scenarios. In this section, the scenarios of industrial processes suitable for NB-IoT based monitoring are introduced.

6.4.1 The massive device support for dense production

The internet of things is a technology, which considers support of massive number of devices. The dense production where the industrial equipment is placed compact in a confined area implies multiple sensors, which send data about their condition to some monitoring center periodically.

The NB-IoT technology enable to perform monitoring of massive number of devices. Therefore, for the scenario of dense production, which is ran by ABB equipment and monitored by the NETA-21 tool, a use case of NB-IoT technology has been developed. The scheme of the use case is shown on Figure 19.

In the use case, the industrial process is ran by electrical motors. The motors are controlled by the ABB industrial drives. The NETA-21 monitoring tool collects data from the connected industrial drives and transfer the information to a cloud via the NB-IoT system.

6.4.2 The extended coverage and battery lifetime for sparse production

A production process is not necessary limited by some area. It could consist of several distant parts, which are anyway connected and require the same monitoring process. In addition, the remote parts of the process could be placed in difficult to access area, where human maintenance is complicated.

The advantages of cellular network as well as extended coverage and battery lifetime of NB-IoT technology allow to apply NB-IoT transmission for the scenario where in addition to the main production area where are another production points,

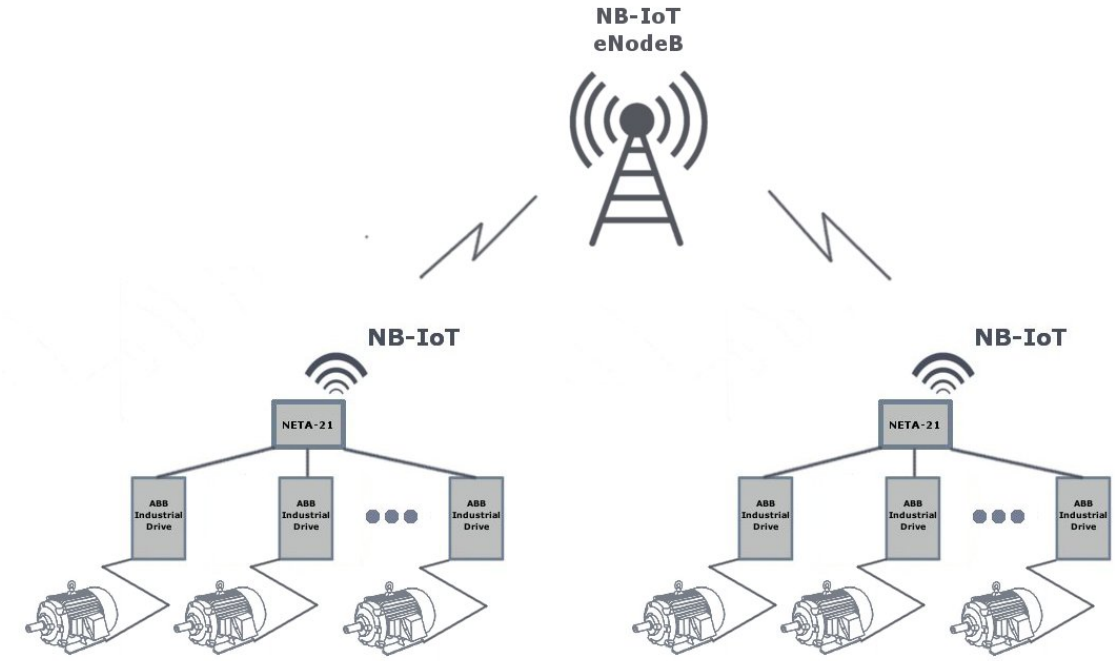


Figure 19: The scheme of the dense production use case. (Courtesy T. Kanerva)

which could be located in rural areas with bad coverage of mobile network. The scheme of the scenario is presented in Figure 20.

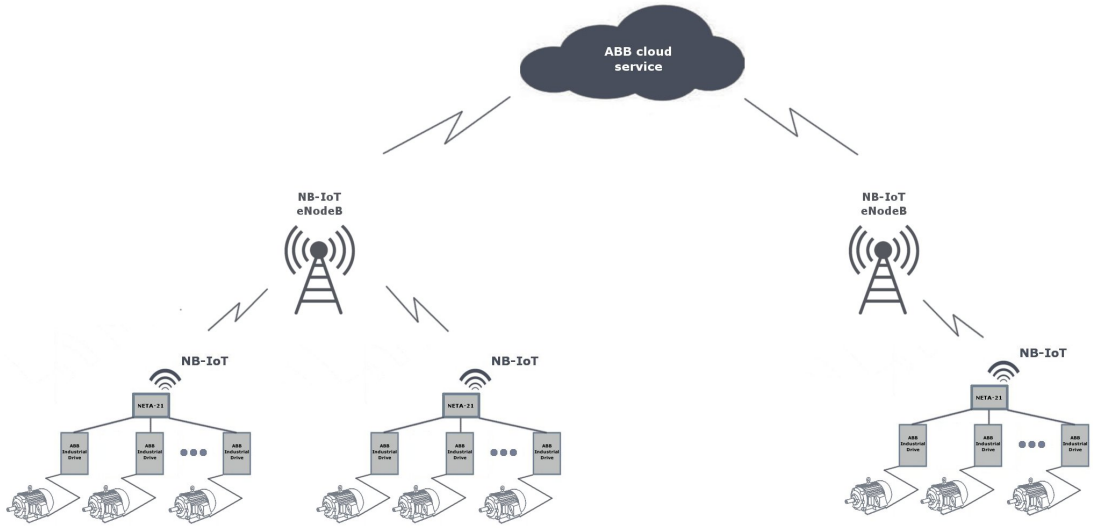


Figure 20: The scheme of the sparse production use case. (Courtesy T. Kanerva)

6.4.3 Cellular technology for mobile production

The NB-IoT has the advantages of a cellular technology and licensed frequency band. Thereby, it can be used for transferring monitoring data from mobile stations, which

use ABB equipment. After the mobile station is deployed at the new place, the data from the industrial drive could be sent to the nearest Base Station (BS) without any additional configurations.

The mobile stations could be:

– equipment for building, which is deployed every time in different construction areas;

– marine electrical equipment, which is placed on vessels and platforms;

– electrical equipment used for mineral resource industry and geological works;

– equipment used in power generation, etc.

The scheme of usage of NB-IoT for mobile production is shown on Figure 21.

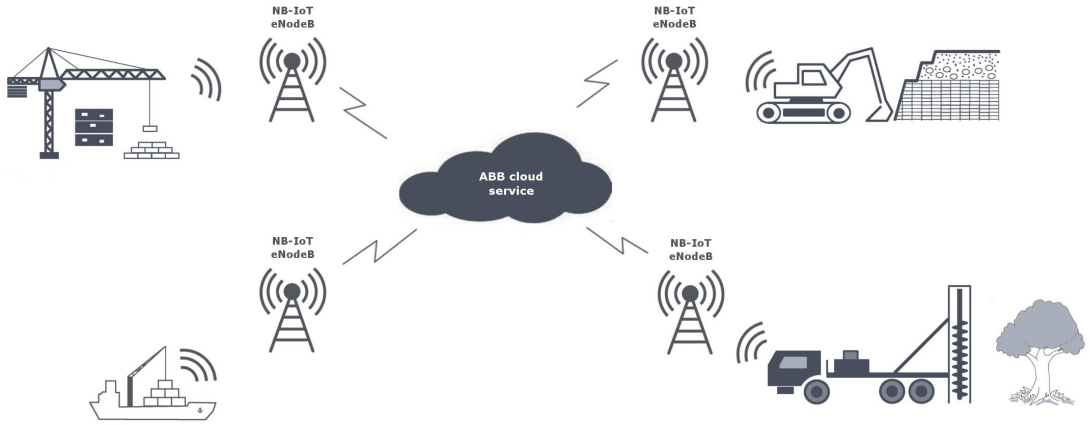


Figure 21: The scheme of the mobile production use case. (Courtesy T. Kanerva)

7 Summary

In this work, the application of the NB-IoT technology for industrial process monitoring was studied. The considered industrial process is a production phase ran by an electrical motor. In turn, the operation of the electrical motor is controlled by an industrial drive. Finally, the performance parameters of the industrial drive are collected by a NETA-21 monitoring tool and transferred to a cloud, where this data is analysed for the purpose of the industrial process optimization. A part of the interest in this process is the transmission of the collected parameters from the NETA-21 monitoring tool to the cloud. This communication can be performed with the NB-IoT technology. NB-IoT is a new cellular technology, which is suitable for specific MTC. In this MTC, stationary (non-mobile) NB-IoT UEs are placed in hard-to-access points and they send small packets of data. Typically, in this case, the transmission is not delay sensitive.

At first, to perform the study, a literature survey was done covering the major technologies by the thesis topic. The Background section (Section 3) contains the results of the literature search and describes the fundamentals of the technologies and systems, which have been studied in this thesis.

The second part of the thesis is the searching of information about NB-IoT chipset modules. The NB-IoT modules can be connected to NETA-21 to perform transmission of the collected data with NB-IoT technology. As there is no mass production of the NB-IoT chipsets yet, the information was obtained by a questionnaire for chipset producers. The questionnaire is presented in Section 4.1 and the results of the questionnaire are described in Section 6.1. The NB-IoT chipset producers forecast that the chipsets will be available in the turn of the 2017 year – in the beginning of the 2018 year. The price of a chipset, estimated by a producer, will be higher than the generally defined: the price range is 7–10 $, while the generally established price is 5$. The value chain of the chipset production is horizontal. The Quectel company provides eSim support, GPS/GNSS support, the application processors integration, and the application development platform. The Altair company could provide information only about the supported application development platform.

After collecting information about the technologies and equipment, the NB-IoT for industrial drive data transmission was tested and validated. The transmission was performed with the ARF developed by the Department of Communications and Networking. In this thesis, two different sets were build up for this purpose. The first set is a NETA-21 ARF set that performs NB-IoT measurement of transmission of data, which have been randomly generated on NETA-21 monitoring tool. The set details and the measurement results are presented in Sections 5.1 and 6.2, respectively. The second set is a demo set, which validates ARF by the NB-IoT transmission of industrial drive parameters. The parameters are read from the industrial drive and then transmitted to the NB-IoT eNodeB with a purpose to represent them with a web interface after the receiving. The set details and transmission results are presented in Sections 5.2 and 6.3, respectively.

The testing of the NB-IoT system for industrial drive data transmission shows that the elapsed time of the transmission is ≈200 ms for the transmission with a

low repetition factor (4–16 repetitions) and ≈1.2 for the transmission with a high repetition factor (128 repetitions). The tested packet size, which is selected to be representative data packet size of an industrial process monitoring, does not affect the system performance significantly. Therefore, for transmission of a typical 1000 bytes packet and containing collected data from an industrial drive, ARF configured with a low repetition factor achieves a data rate of ≈5 kB/s. The data rate of ARF configured with a high repetition factor is ≈0.8 kB/s. These results show that the usage of ARF for industrial driving monitoring is possible though ARF brings limitations that make the monitoring system to support a lower number of industrial drives, or, to decrease data aggregation intervals if the signal is not strong enough. Further development of the ARF will let to achieve better data rate as well as to perform more accurate measurements.

The demo set was shown to work stable and allow to remotely monitor the performance parameters of the industrial drive. With some changes to the demo set, the two-way communication between an industrial drive and the monitoring module can be made possible as future work.

The final part of the thesis is describing the possible use cases of the NB-IoT technology for the monitoring of an industrial process. The use cases were determined according to the specific features of the NB-IoT technology. The details of use cases determination and the description of the use cases can be found in Sections 4.2 and 6.4, respectively. The study of the use cases shows that massive support of NB-IoT technology allows to use it for dense production monitoring. The extended coverage and battery lifetime of NB-IoT suits well for sparse production monitoring. For the last use case, the advantage of cellular technology can be applied for mobile production.

The main conclusions of the work read as follows:

1. NB-IoT technology meets the requirements of NETA-21 monitoring tool data transmission and, therefore, could be used for the purpose of industrial process monitoring.

2. The NB-IoT chipsets will be soon available for mass production that would allow practical testing of NB-IoT technology for industrial drive monitoring.

3. The expected price of an NB-IoT chipset is higher than it has been established (by 3GPP) and, hence, the including of the NB-IoT transmission feature in the monitoring tool will require a careful estimation of the profit.

4. The ARF system at the present condition satisfies the industrial drive data transmission requirements though it brings limitations for a drive monitoring system.

References

[1] M. R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel, and L. Ladid. Internet of things in the 5G era: Enablers, architecture, and business models. *IEEE Journal on Selected Areas in Communications*, 34(3):510–527, March 2016.

[2] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang. A vision of IoT: Applications, Challenges, and Opportunities With China Perspective. *IEEE Internet of Things Journal*, 1(4):349–359, Aug 2014.

[3] L. D. Xu, W. He, and S. Li. Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, 10(4):2233–2243, Nov 2014.

[4] J. Schlienz and D. Raddino. Narrowband Internet of Things. Application note, Rohde&Schwarz, August 2016.

[5] J. Gozalvez. New 3GPP standard for IoT [mobile radio]. *IEEE Vehicular Technology Magazine*, 11(1):14–20, March 2016.

[6] Y. P. E. Wang, X. Lin, A. Adhikary, A. Grovlen, Y. Sui, Y. Blankenship, J. Bergman, and H. S. Razaghi. A primer on 3GPP Narrowband Internet of Things. *IEEE Communications Magazine*, 55(3):117–123, March 2017.

[7] S. Landström, J. Bergström, E. Westerberg, and D. Hammarwall. NB-IoT: A sustainable technology for connecting billions of devices. *Charting the Future of Innovation*, 93(3), April 2016. Ericsson Technology Review.

[8] LTE evolution for IoT connectivity. Nokia, 2017. White paper. Product code SR1702006775EN.

[9] C. Yu, L. Yu, Y. Wu, Y. He, and Q. Lu. Uplink Scheduling and Link Adaptation for Narrowband Internet of Things Systems. *IEEE Access*, 5:1724–1734, 2017.

[10] LTE quick reference: Drx (discontinuous reception) - cdrx (connected mode drx). Retrieved August 13, 2017, from http:\\www.sharetechnote.com\html\Handbook_LTE_DRX.html.

[11] M. Krasnyansky. Universal TUN/TAP device driver, 1999-2000. Retrieved August 20, 2017, from https://www.kernel.org/doc/Documentation/networking/tuntap.txt.

[12] Y. Beyene N. Malm. LPWA NB-IoT Demo, May 2017.

[13] ABB. ABB drives and controls. The green guide to more profitable business.

[14] Firmware manual: ACS880 primary control program. ABB, 2012. ABB industrial drives.

[15] User's manual: ACS880-01 democase. ABB, 2012. ABB industrial drives.

[16] T. Loponen and Z. Kosa. Remote monitoring drives service. ABB, April 2016.

[17] T. Loponen. Personal discussion, February 2017.

[18] V. Jung. How to access NETA-21 first time. ABB, March 2013.

[19] User's manual: NETA-21 remote monitoring tool. ABB, 2014. Remote monitoring options for ABB drives, converters and inverters.

[20] Modbus application protocol specification v1.1b. Modbus-IDA, 2006. http://www.Modbus-IDA.org.

[21] Modbus messaging on TCP/IP implementation guide v1.0b. Modbus-IDA, 2006. http://www.Modbus-IDA.org.

[22] G. Collins. Pymodbus documentation. Release 1.0, April 2017. https://pymodbus.readthedocs.io/en/latest/index.html.

[23] T. Eliseeva. ABB NB-IoT Demo, instruction, July 2017.

[24] Mika J. Karna. Personal discussion, June 2017.

A The chipset producers questionnaire

NB-IoT Chipset Survey

In this questionnaire I would like to ask you several questions about production of Narrowband Internet of Things (NB-IoT) chipsets and NB-IoT external modules (for example, usb modules).

1. Please, enter your company name

Company name:

2. In your view, when will the external modules be available in mass production?

A bulk purchase of 1000 units will be available for buying

- ○ In several months
- ○ In the end of 2017
- ○ In the beginning of 2018
- ○ In the end of 2018
- ○ Later

3. In your view, what will the cost of the first generation external module be for a 1000 pieces bulk purchase?

The cost of the module will be

- ☐ Less then 3 $
- ☐ 3 - 5 $
- ☐ 5 - 7 $
- ☐ 7 - 10 $
- ☐ 10 - 15 $
- ☐ More then 15 $

4. Which value chain, in your view, will be dominant in the second generation of NB-IoT products?

The value chain will be

- ○ Vertical integration: chipset manufactures produce also final products (sensors).
- ○ Horizontal integration: chipset manufactures, module producers and module integrators are independent.

5. In your view, how will be eSIM support for NB-IoT modules?

Is your company going to support eSIM for 2018 year NB-IoT module product?

- ○ Yes, eSIM will be integrated in module.
- ○ Yes, eSIM will be supported but it will be a separated module.
- ○ No, eSIM will not be supported.

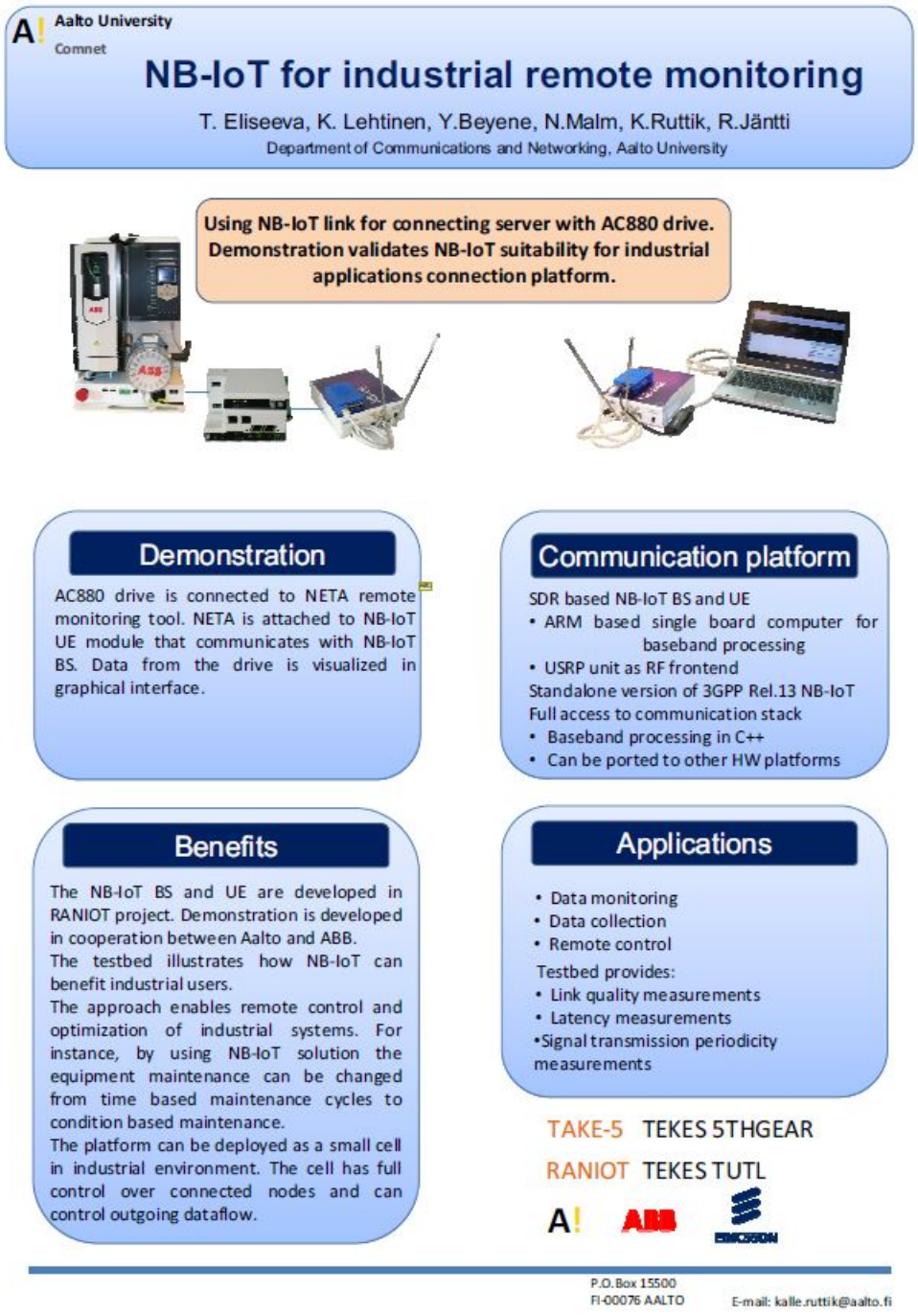
6. What will be the feature set of NB-IoT chipsets, which will be produced in your company?

The feature set will include

- ☐ Application processors integration
- ☐ Application development platform availability
- ☐ Other

7. Thank you for your answers. If you have any comments, please, write them below.

B The ABB demo set poster



C LWPA demo instructions

LPWA NB-IoT Demo

Template and base: Nicolas Malm
Technical information review: Yihenew Beyene

Created: 19/05/2017
Updated: 22/05/2017

Contents

Chapter 1

Description

The LPWA 2017 demo code demonstrates a partial implementation of NB-IoT Release 13 used to provide data transfer to a single UE connected to a single eNodeB. Data is read by the UE from sensors and sent as an IP–based payload to the eNodeB for display on a Web-based user interface. All communication features are implemented as a C++14 software-defined radio and execute in real time on an ARMv8-based Odroid C2.

Chapter 2

Components

Components, both hardware and software, are presented in this section.

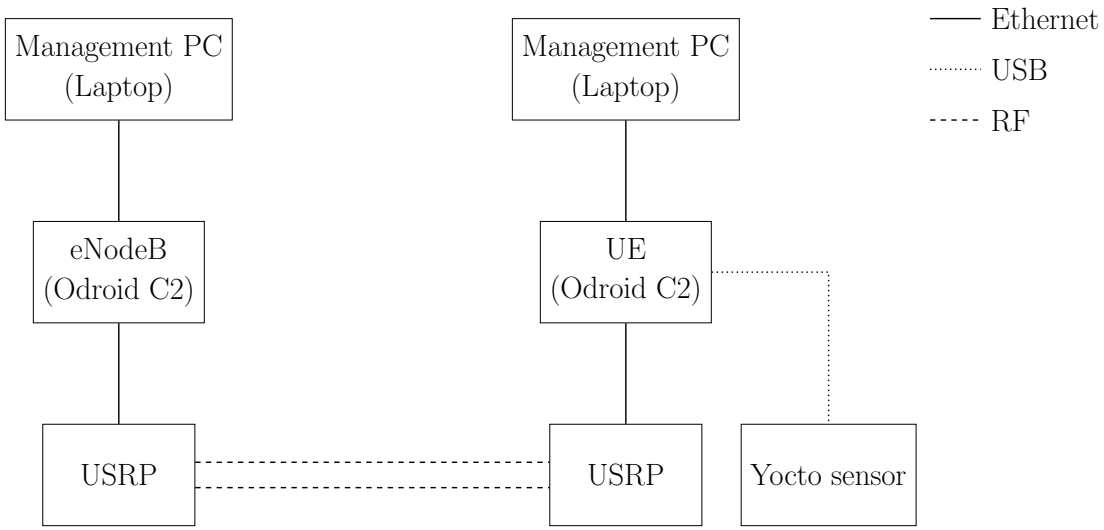
2.1 Hardware

The demo is comprised of the following hardware components (quantity in parentheses):

- Network-capable USRP (2)
- Odroid C2 (2)
- Yocto temperature/humidity sensor (1)
- Antennas **OR** RF cables and attenuators (2)
- Laptop (2)
- USB-Ethernet adaptors (2)
- Cat5e+ Ethernet cables (4)

Note: The above list does not include required power supplies.

The laptops listed above serve a dual purpose. Firstly, they are used to connect to the via Odroids via the USB-Ethernet adaptors. The second function is to provide a display for the UI at the eNodeB.



2.2 Software

The demo is comprised of the following software components:

- NB-IoT eNodeB binary
- NB-IoT UE binary
- Sensor reader Python script
- HTTP-server Python script
- Javascript UI script

Chapter 3

Operation

Setup of the demo is described in Section 3.3. This may need to be adapted to the equipment used. Section 3.2 presents the supported configuration parameters. These can be used to improve performance or troubleshoot problems.

3.1 Principle of Operation

Upon start-up, the UE will synchronise to the eNodeB. Once this is done, the two establish a data connection. The latter operates by sending actual data packets if available and dummy packets otherwise. Failed transmissions are re-attempted.

The data sent by the UE is read from its configured TUN interface. This IP traffic is generated by a Python sensor reading script, which sends it using UDP. Values representing the readings are formatted as bytes before being sent.

Once received by the eNodeB, IP and UDP headers are stripped from the data. The payload is then retransmitted using UDP to localhost, where the server script listens for it. Said script also provides an HTTP server to which one should connect to using a browser. Inside the index.html page resides a Javascript script which regularly fetches new values and updates the display using them.

3.2 Configuration

The configuration presented in this section has been successfully tested. Others may work as well. Adapt address, ports, identifiers and other values to the ones present on the equipment used.

3.2.1 eNodeB

The configuration parameters below must be set correctly in order for the eNodeB to operate.

- Inside `nbioteNodeBL1L2L3Test.conf`, set `tun_interface_enodeb` to the appropriate TUN interface (see Section 3.2.3)

3.2.2 UE

The configuration parameters below must be set correctly in order for the UE to operate.

- Inside `nbiotUEL1L2L3Test.conf`, set `tun_interface_ue` to the appropriate TUN interface (see Section 3.2.3)

3.2.3 OS

Ensure that the laptops have the USB-Ethernet adapters configured with IP address `172.16.1.1` with CIDR mask of /30 or smaller. The Odroid is configured to use `172.16.1.2/24`. The built-in Ethernet on the Odroids are configured for `192.168.10.1/24`.
TUN interfaces must be configured in order to access the dataplane of the NB-IoT stack. The steps required and examples commands are listed below:

- Create TUN interface
- Add a routing table entry pointing to the interface
- Set the interface name in UE/eNodeB configuration

The above can be achieved using:

```
ip tuntap add mode tun name tun0
ip link set up dev tun0
ip route add 10.0.0.2 dev tun0
```

After this, you should be able to ping the dataplane using the address routed to the interface.

3.3 Setup Procedure

Getting the demo setup requires at most the steps below. Some may be skipped if no changes have occured since last use. For example, one need not recompile the code every time the demo is run if no changes have been made to the code.

1. Connect Odroids with laptops using the USB-Ethernet adaptors
2. Connect USRPs to eNodeB and UE Odroid built-in Ethernet ports
3. Install antennas or connect cables and attenuators between the USRPs
4. Connect Yocto sensor to UE Odroid
5. Plug in power supplies
6. SSH into the Odroids. On both nodes, open two sessions (one for the NB-IoT binary and one for the application).
7. Check connection with USRPs (e.g.: `uhd_find_devices`, `ping -c 1 -W 1 192.168.10.2`)
8. Ensure TUN interfaces exist on both nodes (`ip addr`). If not, run `configure_tun.sh` in `/home/odroid/`
9. Open `set_time.sh` in `/home/odroid/` in a text editor and change to the correct datetime. Then run the script.
10. Change directory to `Projects/NB-IoT/build` in one SSH session on both nodes
11. Change directory toNB-IoT-Demo2017/UE/sensorApp or NB-IoT-Demo2017/eNB in the other SSH session on both nodes
12. Run `apps/testNbioteNodeBL1L2L3` and `apps/testNbiotUEL1L2L3` in the `Projects/NB-IoT/build` sessions
13. Run `python eNB-sensor-app.py` and `sudo su`, then `python UE-sensor-app.py` in the `NB-IoT-Demo2017/` sessions
14. Connect to the HTTP server from the laptop connected to the eNodeB (by default at `http://172.16.1.2:8888`) using a browser

Chapter 4

Troubleshooting

Known issues with the demo are presented in this section. Issues and remedies listed do not follow any particular order.

4.1 HTTP server script refuses to start

Possible Cause
The port used by the script is not always available for re-use immediately after exit.

Remedy
Change the port number in the script (search for server-port) and in the browser's URL bar.

4.2 Synchronization is unstable and leads to stuck dataplane

Possible Cause
Positive tracking mode drift correction do not currently operate correctly.

Remedy
Swap the USRPs between eNodeB and UE to obtain a negative clock drift rate for the UE.

Note: oscillator drift rates depend on a number of factors and should not be assumed constant.

4.3 UE does not synchronise at all

Possible Cause

Received power is too low to allow for reliable synchronisation.

Remedy

Adjust TX and/or RX power incrementally in `apps/nbioteNodeBL1L2L3Test.cpp` or `apps/nbiotUEL1L2L3Test.cpp` and test again. It may be necessary to iteratively search for workable levels.

Note: excessive power levels can be problematic as well.
Caution: care (i.e. attenuators) must be taken when using cables to avoid burning out the receiver.

4.4 Node reports large number of 'L'

Possible Cause

Insufficient processing speed in baseband processing. This is a much too large area to give a definitive and all encompassing answer to. Potential solutions listed below present the most commonly used techniques.

Remedy

Check that no other significant load is running on the device:

```
top
```

Reduce interrupt coalescing time on the NIC port connected to the USRP:

```
ethtool -C eth0 rx-usecs 10
```

Recompile the code in Release mode:

```
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
make -j 4
```

4.5 UE cannot open the Yocto sensor

Possible Cause 1
Insufficient permissions.

Remedy 1
Try again as root.

Possible Cause 2
The device is not found by the kernel.

Remedy 2
Run `lsusb` and for device id `24e0:0018`. If it isn't visible, try to replugging the sensor. You can also try USB ports on different controllers.

4.6 Yocto sensor reader script quits withtout doing anything

Possible Cause
Unknown.

Remedy
Reboot.

4.7 Cannot SSH into Odroids

Possible Cause
Routing is wrong in the laptops.

Remedy
Run `ip route get 172.16.1.2` to check if the out interface is the USB-Ethernet adapter. If it isn't, one reason can be that the Network Manager has loaded another profile for the adapter or none at all. Check that the correct one is active.

D ABB demo instructions

ABB Nb-IoT Demo

Master thesis worker Eliseeva Tatiana

December 18, 2017

1 Introduction

This document contains an instruction, which could help to set up and run ABB NB-IoT demo. The demo shows transmission of industrial drive parameters with developed in Communication and Networking Department implementation of NB-IoT Release 13. Industrial drive is included in ABB democase.

2 The components

For the demo you will need the following hardware components:

1. ABB ACS880-01 democase
2. NETA-21 monitoring tool
3. Switch
4. Network-capable USRP (2)
5. Odroid C2 (2)
6. Antennas OR RF cables and attenuators (2)
7. Laptop (2)
8. USB-Ethernet adapters (2)
9. Cat5e+ Ethernet cables (5)

3 The principles of operation

In this Demo an industrial drive parameters are read and sent through a NETA-21 monitoring tool, which is used as a Modbus/TCP gateway, to a PC. The PC transmits the data with NB-IoT UE application to another PC, where NB-IoT eNodeB application is ran. The transmission of data is performed by USPRs connected to the PCs. The parameters are presented on laptop, which is connected to the eNodeB PC.

Reding from the industrial drive is performed by a python script, which is ran on UE PC. The script reads parameters from the drive and write them with Modbus protocol. The transmitted data is accepted on the eNodeB side by eNodeB python server script, which also provides HHTP server for data presentation.

The NB-IoT system is ran by two binaries: one on UE PC and another on eNodeB PC. More information about NB-IoT system you could fine in [1].

4 Installation

This section contains information about installation and initialization of the demo modules.

4.1 Installation of the ABB democase

1. Connect ABB democase ports **"IO-panel"** and **Control Panel** with the provided Ethernet cable. The connectors are shown on Figure 1. Without this connection the democase will not start work.

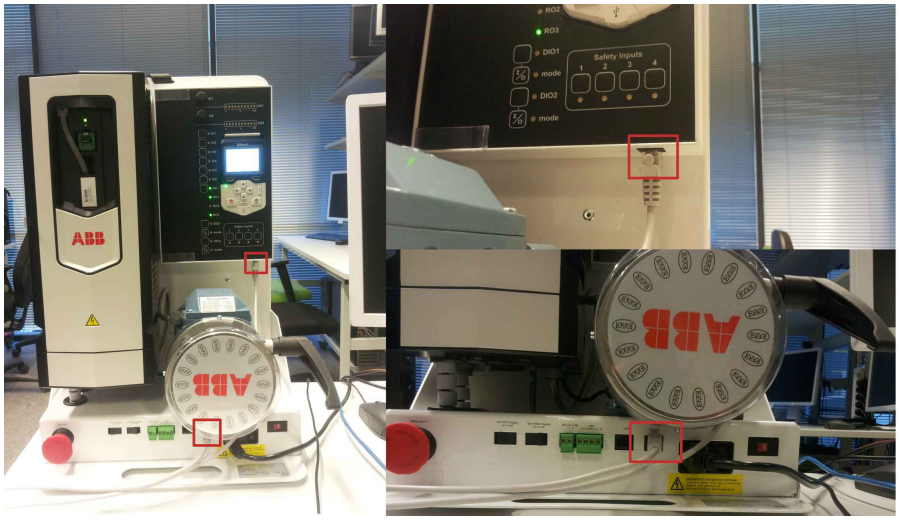


Figure 1: ABB democase installation.

2. Check that the **Input voltage** switch is set to 230V. The switch is marked on Figure 2.
3. Plug in the democase and put button **"O/I"** on. Wait while he democase will turn on.
4. Press the button **DI1** on the democase. It will start working.
5. Set the motor speed with **"AI1"** control knob.
6. Take off the Ethernet cable from **Control Panel** and plug it into the first panel bus connector of NETA-21 monitoring tool (see Figure 3). If you will plug the Democase to the second connector, python script will not be able to read data from the drive because in this case the drive has another address.

4.2 Installation of NETA-21 monitoring tool

To install the NETA-21 monitoring tool you will have to connect it first to the power supply. If you connect the NETA-21 monitoring tool for the first time you should get access to web interface and allow Modbus communication. You could also get ssh access to NETA-21.

4.2.1 Electrical installation

1. Connect the power supply cord to connector **X1** of the NETA-21 monitoring tool (See Figure 4): **red** wire is **Plus** and should be connected to the pin **1**; **black** wire is **Minus** and should be connected to the pin **2**. You could see + and − marks of the pins if you a bit move the connector.

2. Plug in the power supply.

4.2.2 Connection of a PC to the NETA-21 monitoring tool

1. Check that the **STAT** LED on the NETA-21 part (between **PWR** and **MON**) is green.
2. Check that **ETH1** cable is unplugged.
3. Press the **SD RJ45** button for 5 seconds or until the **PC EHT1** LED starts to blink green. NETA-21 starts functioning as a DHCP server via the **ETH1** port.
4. Connect the network cable from PC to the NETA-21 **ETH1** port. NETA-21 provides a dynamic IP to the PC.
5. Check that the **ETH1** LED indicates a connection: green blink – waiting for PC cable connection; green – connected. [2]
6. To get access to the web interface and allow Modbus communication **(NOTE: these step is not needed for the existing demo)**:
   (a) Using the PC web browser navigate to http://192.168.230.1.
   (b) Enter **Username:** admin
   (c) Entet **Password:** Tatiana
   (d) Front page of web interface will open after authorization.
   (e) In web interface choose **Settings** → **Network services** → tab **Services** and in window **Services / firewall settings** enable **All** for **Modbus/TCP gateway**.
   (f) In the catalogue choose **Device interfaces** → tab **Modbus/TCP gateway**.
   (g) Click on the device and in **Device parameters** change **Read-write flag** to *Read-write*.
7. To get ssh access to the NETA-21 monitoring tool **(NOTE: these step is not needed for the existing demo)**:
   (a) Open a terminat window
   (b) Enter command: `ssh factory@192.168.230.1`
   (c) Enter **Password:** Bj6X!4K5BC
   (d) To get root password enter command `su` and **password** e2C@3UJhwg

4.3 Installation of PCs and USPRs

The installation scheme is in Figure 5.

1. Connect UE Odroid to the switch using USB-Ethernet adapter.
2. Connect NETA-21 monitoring tool and the laptop to the switch.
3. Connect eNodeB Odroid to the laptop using USB-Ethernet adaptor.
4. Connect the USPRs to the Odroids using build-in Ethernet port.
5. Plug in power supplies.
6. Set IP addresses for laptops: 192.168.230.4/24 and 192.168.230.5/24.
7. SSH to the Odroids (`ssh odroid@192.168.230.2` for UE Odroid and `ssh odroid@192.168.230.3` for eNodeB Odroid). Use two sessions: one for NB-IoT work another for python applications.
8. Check connection to USPRs with command `uhd_find_devices`.
9. Open file *configure.sh* and set there correct date and time.
10. Set correct time and set up TUN interfaces by running `sudo ./configure.sh` in `home/odroid`.
11. In the first session change directory to `cd Projects/master/build` on both Odroids.
12. Run UE and eNodeb NB-IoT applications with `./apps/testNbioteNodeBL1L2L3` and `./apps/testNbiotUEL1L2L3`.
13. In the second session change directory to `/Documents/Modbus/`.
14. Run UE and eNodeB python applications with `python UE-ModbusClient-app.py` and `python eNB-ModbusClient-app.py`.
15. Connect to the HTTP server from the eNodeB laptop with `http://192.168.230.3:8888` using a browser. [1]

5 Troubleshooting

5.1 UE-ModbusClient-app.py script cannot read the parameters

This means that NETA-21 does not respond. You should reboot the NETA-21 monitoring tool.

5.2 eNodeB starts, UE synchronizes, but eNodeB does not accept packets or accepts just a few

This is an error of USPR. Reboot USPRs.

5.3 eNodeB-ModbusClient-app.py does not start with error `<class 'socket.error'>`

This means that port is busy. run the python script with different port number, for example, `python eNB-ModbusClient-app.py -p 9999`

References

[1] Nicolas Malm, Yihenew Beyene. *LPWA NB-IoT Demo*. Department of Communications and Networking, Aalto University, 1993, May.

[2] Vassili Jung. *How to access NETA-21 first time*. ABB, 2013, March.

Figure 2: ABB democase installation.

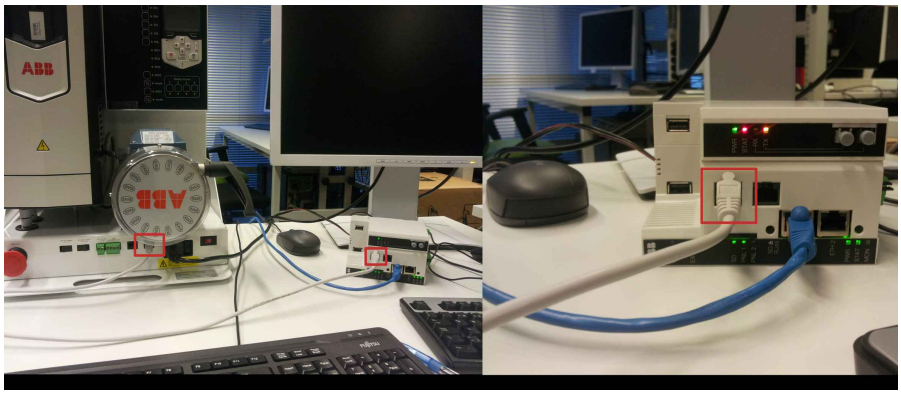


Figure 3: ABB democase and NETA-21 connection.

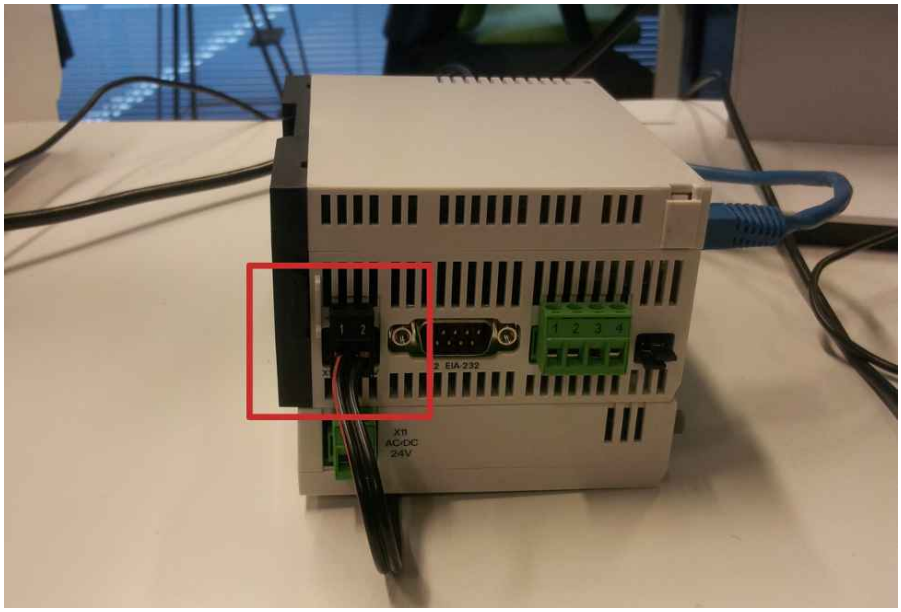


Figure 4: NETA-21 electrical installation.

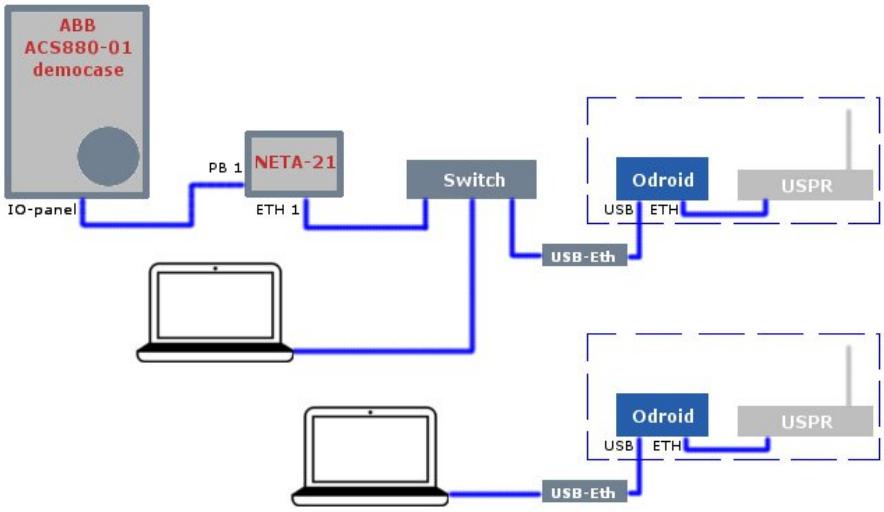


Figure 5: Installation Scheme.

E NETA-Ps pseudocode

```
1   UDPClient
2
3       Import Data class;
4       Import modules socket, time and math;
5
6       Load the initial data from the configuration file:
7       - port_number;
8       - host_address;
9       - number_of_drives;
10      - number_of_iterations;
11      - waiting_interval;
12
13      Create client_socket(host_address, port_number);
14
15      While (current_iteration_number < number_of_iterations)
16          Create_data_packet(number_of_drives, current_iteration_number,
            number_of_iterations) with Data class;
17          Record start_time;
18          Send data packet to the server;
19          Get acknowledgment from the server;
20          Record stop_time;
21          Calculate and log round_trip_time = stop_time - start_time;
22          Sleep(waiting_interval);
23
24      Close the client_socket;
25
26
27  class Data
28
29      Import module random;
30
31      Define size_of_packet_from_a_single_drive = 100;
32
33      Create_data_packet(number_of_drives, current_packet_number,
        number_of_packets)
34          Define packet_size = number_of_drives *
            size_of_packet_from_a_single_drive;
35
36          Define data_packet[1] = current_packet_number;
37          Define data_packet[2] = number_of_packets;
38          Define data_packet[3, packet_size] = random char;
39
40          return data_packet;
41
42
43  UDPServer
44
45      Import module socket;
46
47      Load the initial data from the configuration file:
48      - port_number;
49
50      Create server_socket
51
52      Bind server_socket to the port_number;
53
54      While true
55          Recover data_packet and client_address;
56          Send an acknowledgment to client_address;
57          Push data_packet to TUN;
58
59          If data_packet[1] == data_packet[2]
60              close server_socket;
61              break;
62
```

F UE-Ps pseudocode

```
UE-ModbusClient

    Import modules OptionParser, socket, random, time, re, subprocess,
    sys, select, tty, termios, Thread, struct;
    Import ModbusTcpClient from pymodbus.client.sync;

    Define global variables:
        DRIVE_ERROR;
        SPEED_STR;
        FREQUENCY_STR;
        CURRENT_STR;
        TORQUE_STR;

    Initialize DRIVE_ERROR = false;


    class ModbusABBClient(Thread)
        Define keep_running = true
        Create modbus_client = ModbusTcpClient(NETA-21_address =
        '192.168.230.1', port = 502);

        Try
            If modbus_client can be connected
                While keep_running
                    Try
                        # Read motor speed
                        Read speed_value = modbus_client
                            .read_holding_registers(parameter_id =
                            100, group_id = 1, drive_id =
                            33).registers[0];
                        SPEED_STR = convert_value_to_char(speed_value);

                        # Read output frequency
                        Read frequency_value = modbus_client
                            .read_holding_registers(parameter_id =
                            105, group_id = 1, drive_id =
                            33).registers[0];
                        FREQUENCY_STR =
                        convert_value_to_char(frequency_value);

                        #Read motor current
                        Read current_value = modbus_client
                            .read_holding_registers(parameter_id =
                            106, group_id = 1, drive_id =
                            33).registers[0];
                        CURRENT_STR =
                        convert_value_to_char(current_value);

                        #Read motor torque
                        Read torque_value = modbus_client
                            .read_holding_registers(parameter_id =
                            109, group_id = 1, drive_id =
                            33).registers[0];
                        TORQUE_STR =
                        convert_value_to_char(torque_value);

                    Except
                        Sleep(5 seconds);
                        Reconnect modbus_client;
            Else
                Log("Modbus connection failed to open");
            Close modbus_client;
        Except
            DRIVE_ERROR = true;
```

```
56      class sensorStreamer
57
58          Start ModbusABBClient;
59          Define keep_running = true;
60
61          Try
62              Define sleep_time = 3 seconds;
63
64              While keep_running
65                  If DRIVE_ERROR = true
66                      keep_running = false;
67                      ModbusABBClient.keep_running = false;
68                      break;
69
70                  # Create a data packet
71                  data_packet = SPEED_STR + FREQUENCY_STR + CURRENT_STR
                    + TORQUE_STR;
72
73                  push data_packet to TUN;
74          Except
75              Stop all threats;
```

G eNB-Ps pseudocode

```
1   eNB-ModbusClient
2
3       Import modules socket, os, time, sys, posixpath, BaseHTTPServer,
        urllib, cgi, shutil, Thread, mimetypes, StringIO, SocketServer,
        OptionParser;
4
5       Define global variables:
6           NODE_DATA;
7           ul_socket;
8           dl_socket;
9
10      Create ul_socket(AF_INET, SOCK_DGRAM);
11      Create dl_socket(AF_INET, SOCK_DGRAM);
12
13      Initialize NODE_DATA as array of zeros;
14
15      Define a function convert_byte_string_to_short_int(data_string),
        which recovers 16-bit positive number from a string of 2 characters;
16
17      class
        SimpleHTTPRequestHandler(BaseHTTPServer.BaseHTTPRequestHandler)
18
19          The class has been implemented for the previous demo set and
            has not been changed during this Master's thesis.
20
21          The class is a simple HTTP request handler with GET and HEAD
            commands.
22
23      class SensorDataSink
24
25          While true
26
27              Try
28                  Define data = recover from ul_socket;
29
30              Except
31                  Set NODE_DATA to 0;
32                  continue;
33
34              Define unit_id = data[0];
35
36              Define speed_value =
                convert_byte_string_to_short_int(data[1:3]);
37              Apply sign and scale to speed_value;
38
39              Define frequency_value =
                convert_byte_string_to_short_int(data[3:5]);
40              Apply sign and scale to frequency_value;
41
42              Define current_value =
                convert_byte_string_to_short_int(data[5:7]);
43              Apply sign and scale to current_value;
44
45              Define torque_value =
                convert_byte_string_to_short_int(data[7:9]);
46              Apply sign and scale to torque_value;
47
48              Define NODE_DATA = [unit_id, speed_value, frequency_value,
                current_value, torque_value];
49
50          Define main function
51
52              Parse from arguments or define server_port;
53
54              Try
55                  Define server = SocketServer.TCPServer(server_port,
                    SimpleHTTPRequestHandler);
56                  Bind ul_socket;
```

```
57                      Start SensorDataSink;
58
59                  Except
60                      Stop all threads;
```